Titre: Title:	Approches hybrides pour des problèmes intégrés d'ordonnancement et de routage de véhicules sans conflits
Auteur: Author:	Ayoub Insa Corréa
Date:	2005
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Corréa, A. I. (2005). Approches hybrides pour des problèmes intégrés d'ordonnancement et de routage de véhicules sans conflits [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7553/

UNIVERSITÉ DE MONTRÉAL

APPROCHES HYBRIDES POUR DES PROBLÈMES INTÉGRÉS D'ORDONNANCEMENT ET DE ROUTAGE DE VÉHICULES SANS CONFLITS

AYOUB INSA CORRÉA
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR (Ph.D.)
(MATHÉMATIQUES DE L'INGÉNIEUR)
DÉCEMBRE 2005

Your file Votre référence
ISBN: 978-0-494-16988-9
Our file Notre référence
ISBN: 978-0-494-16988-9

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

APPROCHES HYBRIDES POUR DES PROBLÈMES INTÉGRÉS
D'ORDONNANCEMENT ET DE ROUTAGE DE VÉHICULES SANS CONFLITS

présentée par : CORRÉA Ayoub Insa
en vue de l'obtention du diplôme de : Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de :

M. DESAULNIERS Guy, Ph.D., président
M. LANGEVIN André, Ph.D., membre et directeur de recherche
M. ROUSSEAU Louis-Martin, Ph.D., membre et codirecteur de recherche
M. GAMACHE Michel, ing., Ph.D., membre
M. RENAUD Jacques, Ph.D., membre externe

À ma fille Awa Dieumbe, Mariam, ma Grand-mère maternelle, mes parents et mes proches. Ceci est la quête d'un homme qui trace son chemin en jetant des ponts entre deux mondes différents.

REMERCIEMENTS

Je tiens à exprimer ma gratitude à :

- André Langevin pour son soutien intellectuel, moral et financier. Il a encouragé en moi la patience, l'humilité, la persévérance dans l'effort, l'esprit d'initiative et le sens de l'autonomie. Il a été toujours présent pendant les bons et mauvais moments que j'ai vécu. Il SAIT...À sa conjointe Gisèle pour sa gentillesse.
- Louis-Martin Rousseau pour la qualité de nos discussions et son soutien financier. C'est une grande qualité d'être au bon endroit au bon moment.
- Diane Riopel pour ses grandes idées et son soutien moral.
- Mme Burney-Vincent pour sa confiance en moi et ses conseils en pédagogie.
- Bernard Cheung pour sa sagesse.
- Nathalie Marcoux pour son dévouement.
- Nathalie Perrier, Alberto, Serge et Djenet pour leur compagnonage et leur amitié.
- Chacun des membres du groupe Polygistique.
- Benoît Forest, Paul Brunelle pour leur support informatique et leurs très agréables conversations.
- Marielos Jimenez, Evelyne Rousseau, Francine Benoit, Diane Bernier et Suzanne Guindon, je leur dis bravo et merci pour leur aide !
- Aux membres de notre équipe de soccer formée de professeurs, associés de recherche et étudiants.

RÉSUMÉ

L'ordonnancement et le routage intégrés sans conflits de chariots constituent des tâches complexes dans les contextes d'un atelier flexible et d'une mine souterraine. Les buts visés par l'intégration de l'ordonnancement et le routage intégrés sans conflits de chariots sont le dimensionnement de la flotte de chariots, leurs mouvements en toute sécurité et la prise en compte de contraintes réalistes comme l'orientation appropriée des chariots. Pour gérer l'atelier flexible et la mine souterraine d'une façon réaliste, il convient d'intégrer au moins deux niveaux d'optimisation : l'ordonnancement des tâches de collecte/livraison et le routage sans conflits des chariots.

Dans cette thèse, trois algorithmes hybrides sont proposés pour résoudre de façon exacte des problèmes intégrés d'ordonnancement et de routage sans conflits. Les deux premiers problèmes sont similaires et sont étudiés dans le contexte d'un atelier flexible tandis que le troisième problème est étudié dans le contexte d'une mine souterraine. Chacun des trois articles de cette thèse présente un algorithme de décomposition hybride qui combine la programmation par contraintes (PC) et la programmation linéaire en nombres entiers (PLNE). Notre approche globale décompose chaque problème en deux parties : la première consiste à ordonnancer les tâches tandis que la seconde s'occupe du routage sans conflits des chariots. La partie ordonnancement est résolue grâce à la PC tandis que la partie routage sans conflits est résolue par la PLNE.

La première méthode de décomposition de cette thèse porte sur des travaux préliminaires qui sont à la base des deux algorithmes présentés dans les deux articles subséquents. Le premier algorithme présenté a un défaut majeur : trop de solutions optimales pour le modèle de PC mais irréalisables pour le modèle de PLNE sont générées.

Dans la deuxième méthode de décomposition, des contraintes plus réalistes sont prises en compte et aussi bien le modèle de PC que le modèle de PLNE sont améliorés. Des

coupes logiques de réalisabilité (de type *nogood*) sont ainsi générées. L'algorithme qui en résulte permet de résoudre des problèmes contenant jusqu'à six chariots. Toutefois, cet algorithme est limité par la nature très disjonctive des coupes de réalisabilité qui sont générées au cours des itérations.

Dans la troisième méthode de décomposition, les coupes de réalisabilité sont abandonnées au profit d'une meilleure analyse du problème (l'algorithme conçu est basé sur la connaissance du réseau utilisé). Le modèle de PC est totalement remanié pour faire un traitement préventif des conflits. L'analyse du problème permet aussi de générer de meilleures bornes inférieures pour une convergence rapide du modèle de PC. Le modèle de PLNE assure le traitement exact de l'orientation des chariots. Les réseaux considérés sont en forme d'arbre et les tests sont faits sur ces trois réseaux pour analyser des scénarios de croissance de la mine souterraine. Avec notre approche, la non disponibilité totale ou partielle de galeries causée par certains évènements (bris de chariots, chutes de roches, fuites d'eau par exemple) peut être prise en compte. Des problèmes contenant jusqu'à six chariots ont été résolus. Notre méthode constitue un bon outil de dimensionnement de la flotte de chariots dans les contextes de l'atelier flexible et de la mine souterraine.

ABSTRACT

The integrated scheduling and conflict free routing of vehicles are complex tasks in the contexts of a flexible manufacturing system (FMS) and an underground mine. The goals of the integrated scheduling and conflict free routing of vehicles are fleet sizing, safe movements of vehicles while taking into account realistic constraints such as the physical orientation of vehicles. To manage the FMS and the underground mine, it is necessary to integrate at least two levels of optimization : the scheduling of pick-up/delivery tasks and the conflict free routing of vehicles.

In this thesis, three hybrid algorithms are proposed to solve exactly integrated problems of scheduling and conflict free routing of vehicles. The first two problems are similar and are addressed in the context of a FMS while the third one is solved in the context of an underground mine. Each of the three decomposition methods of this thesis presents a hybrid decomposition algorithm combining Constraint Programming (CP) and Mixed Integer Linear Programming (MILP). Globally, our approach decomposes each problem in two components : the first component consists of scheduling tasks while the second one solves the conflict free routing of vehicles.

The first decomposition method consists of some preliminary results which are starting points of the two algorithms presented in the subsequent papers. The hybrid algorithm presented has a major limitation : too much optimal solutions of the CP model but unfeasible for the MILP model are generated.

In the second decomposition method, more realistic constraints are taken into account. The CP model as well as the MILP model are enhanced. Logic feasibility cuts (*nogood* type) are generated. The resulting hybrid algorithm enables to solve instances with up to six vehicles. However, the algorithm is hindered by the highly disjunctive nature of the logic feasibility cuts generated during the iterations.

In the third decomposition method, the logic feasibility cuts are dropped and we profit from a better analysis of the problem (the designed algorithm is based on the knowledge of the networks at hand). The new CP model allows for a preventive treatment of conflicts. Furthermore, the analysis of the problem enables to generate better lower bounds for a fast convergence of the CP model. The MILP model ensures the exact treatment of physical orientation for the vehicles. The three networks used are tree-shaped and the tests are done to analyze some growth scenarios of the underground mine. With our approach, the partial or total non availability of mine roads (due to some events like vehicles breakdowns, rock falls, water leakage, etc.) can be taken into account. This decomposition method is a good tool for fleet sizing in the contexts of a FMS or an underground mine.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ANNEXES

CHAPITRE 1 : INTRODUCTION

Le pilotage de chariots est une tâche très complexe dans les contextes d'un atelier flexible et d'une mine souterraine. Des tâches de collecte et livraison doivent être exécutées par des chariots en des endroits précis. Une tâche signifie une collecte ou une livraison de palettes dans le contexte de l'atelier flexible et chargement ou déchargement de minerai dans le contexte de la mine souterraine. Une requête de transport est constituée d'une collecte et de la livraison associée. Dans le contexte de l'atelier flexible, une estimation du temps au plus tôt de début de chaque collecte ou livraison est disponible et notre objectif consiste à minimiser les retards de livraison. Par contre dans le contexte de la mine souterraine, aucune estimation du temps de début au plus tôt des tâches de chargement ou déchargement n'est disponible et notre objectif consiste à minimiser le temps de fin de la dernière livraison.

Cette thèse porte sur la modélisation et la résolution de problèmes intégrés d'ordonnancement et de routage sans conflits de chariots. Certaines considérations font que les problèmes intégrés d'ordonnancement et de routage sans conflits de chariots sont très difficiles à résoudre :

(1) Toute décision dans la partie ordonnancement des tâches a un impact sur la partie routage sans conflits des chariots, et *vice versa*. Il est important de savoir quel chariot doit exécuter quelle tâche et à quel moment de l'horizon de planification. Un chariot peut être inutilisé durant tout l'horizon mais peut se déplacer pour céder la place à un autre chariot occupé par l'exécution d'une tâche.

(2) Dans la mesure où tous les chariots se partagent le même réseau, des collisions sont possibles et doivent être évitées (à la différence d'un problème de tournées de véhicules classiques où une ville peut être visitée en même par deux véhicules différents). Aussi un chariot peut visiter plusieurs fois le même point de tâche.

(3) Il existe des contraintes de priorité qui permettent de satisfaire la séquence de production. Par exemple dans le contexte de l'atelier flexible, si deux tâches sont liées par des contraintes de priorité alors aucune autre tâche ne peut être exécutée entre ces deux tâches au même point d'exécution. Aussi si une collecte précède une livraison, cela signifie qu'un produit doit être collecté pour libérer de l'espace et rendre possible une autre livraison. Si une livraison qui précède une collecte signifie qu'un produit doit être livré dans une station de travail puis la machine localisée dans cette station doit le traiter avant qu'il ne soit disponible pour une autre collecte.

(4) La congestion peut très vite s'avérer être un frein à la productivité. C'est pourquoi il est nécessaire d'avoir un outil de dimensionnement de la flotte de gestion bien adapté.

(5) Dans le cas du routage dans une mine, il faut aussi obtenir des solutions qui présentent une orientation cohérente des chariots.

L'intégration de plusieurs niveaux d'optimisation, la conception de nouveaux algorithmes de décomposition et la modélisation efficace de contraintes logiques sont trois défis auxquels est confrontée la communauté de la recherche opérationnelle. Dans cette thèse, nous nous attaquons d'abord l'intégration de deux niveaux d'optimisation (ordonnancement et routage sans conflits) en résolvant trois problèmes intégrés d'ordonnancement et routage sans conflits de véhicules. Dans chacun de ces problèmes, nous proposons une méthode exacte de décomposition hybride qui combine la programmation par contraintes (PC) et la programmation linéaire en nombres entiers (PLNE). Dans chaque méthode de décomposition, le problème maître (ordonnancement) est modélisé avec la PC tandis que le sous-problème (routage sans conflits) est modélisé avec la PLNE.

La première méthode de décomposition est à la base des développements algorithmiques contenus dans les deux articles subséquents et l'algorithme qui y est présenté est conçu pour des réseaux quelconques. Il présente le principal défaut de générer beaucoup de solutions optimales pour le modèle de PC mais irréalisables pour le modèle de PLNE.

La seconde méthode de décomposition est inspirée du premier algorithme. Des contrain tes additionnelles sont ajoutées, la structure de flux est mieux mise en évidence et des coupes logiques de réalisabilité sont générées pour converger plus vite vers la solution optimale du modèle original. Aussi, l'ordonnancement implicite des tâches aux machines situées aux points de livraison est fait. Toutefois, la nature très disjonctive des coupes logiques de réalisabilité constitue le principal défaut de la méthode de décomposition hybride. En outre, aussi bien le modèle de PC que le modèle de PLNE nécessitent une amélioration pour travailler sur des problèmes de plus grande taille (plus de chariots, de requêtes de transport et un plus grand horizon de planification). Aussi, une section entière est consacrée à ce qui a partiellement (ou pas du tout) fonctionné, ce qui a été riche en enseignements pour la conception du troisième algorithme.

Le troisième algorithme évite certains défauts de la première méthode de décomposition : (1) un prétraitement plus important est fait au cours de la génération automatique de la structure de données, (2) l'algorithme est conçu pour des réseaux spécifiques (en arbre), (3) les modèles de PC et PLNE sont améliorés. Le modèle de PC propose le traitement préventif des conflits dans les galeries se terminant par des points de chargement ou déchargement. Aussi, la troisième méthode de décomposition traite l'orientation exacte des chariots (ce qui a été auparavant très peu étudié dans la littérature).

Le chapitre 1 introduit le problème intégré d'ordonnancement et de routage sans conflits de chariots. Dans le chapitre 2, nous présentons une description de la PC et une revue de littérature des travaux pertinents faits dans les contextes des ateliers flexibles et mines souterraines. Aussi, une revue spécialisée des méthodes hybrides est présentée. Dans le chapitre 3, nous présentons la synthèse de nos travaux et nos contributions. Dans le chapitre 4 nous présentons deux méthodes de décomposition qui portent sur deux problèmes similaires dans le contexte d'un atelier flexible. Dans le chapitre 5, nous présentons les développements algorithmiques pour résoudre des problèmes dans le contexte d'une mine souterraine. Dans cette partie, nous faisons une analyse des scénarios d'agrandissements de la mine souterraine. Dans le chapitre 6, nous présentons la conclusion générale et nous énumérons des travaux futurs possibles.

CHAPITRE 2 : REVUE DE LITTÉRATURE

Dans ce chapitre, une revue de littérature est présentée sur les approches exactes concernant les problèmes intégrés d'ordonnancement et de routage sans conflits dans les contextes des ateliers flexibles et des mines souterraines (section 2.1) et les méthodes hybrides de la PC et de la PLNE (section 2.3).

Nous avons jugé utile, pour plus de clarté, de faire précéder cette section d'une description de la programmation par contraintes (section 2.2). La section 2.1 porte aussi sur des problèmes apparentés mais qui traitent la notion d'orientation des chariots. La section 2.3 sur les méthodes hybrides de la PC et de la PLNE est orientée vers les méthodes hybrides de décomposition car les trois méthodes de décomposition présentées dans cette thèse portent sur ce type de méthodes.

2.1. Problèmes d'ordonnancement et de routage sans conflits de chariots dans les ateliers flexibles et les mines souterraines

Cette section présente une revue de littérature orientée vers les problèmes d'ordonnancement et de routage sans conflits de chariots dans les contextes d'un atelier flexible et d'une mine souterraine. Le lecteur est référé à Co et Tanchoco (1991), King et Wilson (1991), Ganesharajah *et al.* (1998) et Qiu *et al.* (2002) pour une revue générale sur les chariots automatiques. Il faut mentionner que peu de travaux ont été réalisés au niveau des méthodes exactes sur les problèmes d'ordonnancement et de routage sans conflits de chariots.

Bien que nos travaux portent sur des approches exactes pour les problèmes d'ordonnancement et de routage sans conflits de chariots, quelques méthodes de résolution heuristiques de ces problèmes sont décrites. La complexité de ce type de problèmes a souvent conduit les chercheurs à développer des méthodes heuristiques plutôt

qu'exactes. Ces heuristiques se limitent souvent à un seul aspect du problème.

C'est ainsi que plusieurs auteurs ont traité le problème de routage sans conflits seulement. Lee *et al.* (1998) présentent une procédure de contrôle en deux étapes pour résoudre le problème de routage sans conflits. Leur méthode heuristique consiste à générer k plus courts chemins dans la première étape. Ensuite dans la deuxième étape, le répartiteur associe un plus court chemin libre à chaque chariot automatique affecté à une requête de transport. Rajotia *et al.* (1998) proposent une stratégie de routage semi-dynamique avec des contraintes sur les fenêtres de temps. Le mouvement des chariots automatiques est basé sur les notions de fenêtres de temps occupées et libres. Krishnamurthy *et al.* (1993) présentent une approche d'optimisation qui consiste à minimiser la durée totale de tous les travaux. Ils supposent que l'affectation des chariots automatiques aux requêtes de transport est déjà faite et ils résolvent le problème de routage sans conflits par une méthode de génération de colonnes. Toutefois, cette méthode de génération de colonnes est utilisée au noeud racine de l'arbre de recherche seulement. Ainsi leur approche n'est pas exacte même si elle donne de très bonnes solutions.

Oboth *et al.* (1999) proposent une heuristique pour résoudre de façon non simultanée un problème de répartition et de routage sans conflits de chariots automatiques. L'ordonnancement des requêtes est fait d'abord puis une heuristique de génération séquentielle de chemins (*Sequential Path Generation* -SPG-) est utilisée pour générer des routes sans conflits. L'heuristique SPG est inspirée de la méthode utilisée par Krishnamurthy *et al.* (1993). Elle est cette fois-ci appliquée à un environnement dynamique avec l'abandon d'hypothèses comme la vitesse constante ou égale de chariots automatiques. Toutefois, il n'y a pas de rétroaction avec le modèle d'ordonnancement quand un conflit survient. Le déplacement du chariot automatique est retardé si une route alternative ne peut être générée. En outre, des règles de positionnement des chariots inutilisés sont appliquées plutôt que de laisser le système les gérer de façon automatique.

Langevin *et al.* (1996) proposent une approche basée sur la programmation dynamique pour résoudre de façon exacte des instances du problème de répartition et de

routage sans conflits contenant deux chariots. Sur un problème similaire, Desaulniers *et al.* (2003) présentent une méthode exacte avec laquelle ils ont fait des tests sur des problèmes comportant jusqu'à quatre chariots. Leur approche combine une heuristique de recherche gloutonne (pour trouver une solution réalisable et fixer des bornes sur les délais), une méthode de génération de colonnes et une procédure de *Branch-and-Cut*. Leur méthode présente toutefois quelques limites car son efficacité dépend beaucoup de l'efficacité de l'heuristique de départ. Si aucune solution réalisable n'est trouvée par l'heuristique de départ alors la solution optimale du problème ne peut être obtenue. En outre, l'heuristique devient moins efficace quand le niveau de congestion augmente.

Dans le contexte des mines souterraines, peu de travaux ont été faits sur les problèmes intégrés d'ordonnancement et de routage sans conflits avec prise en compte de l'orientation des chariots. Quelques auteurs ont traité certains aspects du problème que nous avons étudié dans notre troisième méthode de décomposition. Parmi les rares auteurs qui se sont attaqués à l'orientation des chariots, il faut citer Gamache *et al.* (2004) et Bigras et Gamache (2005) où les problèmes résolus sont en temps réel. Toutefois l'approche de Gamache *et al.* (2004) n'est pas exacte et son graphe ne tient pas totalement en compte l'orientation des chariots sur les arcs d'intersection. Les travaux de Bigras et Gamache (2005) portent essentiellement sur le traitement exact de l'orientation pendant le calcul des plus courts chemins et le routage des chariots. L'orientation des chariots est importante car dans le contexte d'une mine souterraine, le réseau est si exigu qu'un chariot ne peut pas se repositionner une fois arrivé à son point de tâche. Comme Bigras et Gamache (2005), la modélisation de la partie routage orienté sans conflits de notre troisième méthode de décomposition s'inspire des travaux de Krishnamurthy *et al.* (1993) et Vagenas (1991). Ces auteurs ont introduit des modèles qui éclatent les points d'intersection en trois ou quatre noeuds.

Aussi, Beaulieu et Gamache (2004) ont proposé une stratégie globale basée sur la programmation dynamique tenant compte de l'orientation pour résoudre en temps réel des problèmes contenant jusqu'à quatre chariots. Deux réseaux utilisés sur trois contiennent des cycles. Toutefois leur approche donne des résultats plus satisfaisants

lorsque la stratégie chariot par chariot, implantée par Bigras et Gamache (2002), est utilisée en amont. Même avec cette approche en deux étapes, des efforts supplémentaires doivent être faits pour améliorer la qualité de la solution obtenue. Les travaux de Bigras et Gamache (2002), Beaulieu et Gamache (2004) et Gamache *et al.* (2004) sont des approches en temps réel. Toutefois le traitement de l'orientation qu'ils ont fait dépasse le cadre opérationnel : ils ont montré qu'il est possible de construire des réseaux qui assurent une orientation appropriée des chariots. Dans leurs approches, les tâches sont considérées selon leur ordre d'arrivée (donc l'aspect ordonnancement n'apparaît pas) et il faut construire par la suite des routes sans conflits qui donnent une orientation correcte des chariots aux points de tâches.

2.2. Description de la programmation par contraintes

Dans cette section, les éléments nécessaires à une bonne compréhension de la PC sont décrits. Il faut entendre par PC la programmation par contraintes sur les domaines finis (c'est-à-dire que les variables sont à domaine fini). Elle dérive de la programmation logique par contraintes (PLC) et on parle maintenant tout simplement de programmation par contraintes car la PC est actuellement implantée grâce à des langages de programmation structurel et fonctionnel. Contrairement à la programmation linéaire, le terme *programmation* en PC fait référence à un programme informatique. La PC est une technique informatique à l'origine développée indépendamment de la Recherche Opérationnelle (RO) pour résoudre des problèmes très contraints de satisfaction de contraintes. Par la suite, la PC s'est intéressée aux problèmes d'optimisation combinatoire (POC). Les limites de la PC ont conduit au développement de méthodes hybrides de la PC et la RO. Cependant, les frontières entre la PC et la RO deviennent de plus en plus floues.

Pour bien comprendre la PC, il est nécessaire de décrire les concepts de base utilisés dans les problèmes de satisfaction de contraintes (PSC). La PC est une méthode

de résolution pour les PSC et POC. De façon générale, un modèle de PSC est composé d'un ensemble de variables (X), d'un ensemble de domaines (D) et d'un ensemble de contraintes (C) qui spécifient quelles affectations de valeurs dans D à des variables X sont légales. Les notions de base dans tout PSC sont : les variables, les domaines, les contraintes, la réduction (ou filtrage) de domaine, la propagation de contraintes, le branchement, les heuristiques de recherche et enfin l'optimisation. Décrivons maintenant un peu plus en détail ces notions.

2.2.1 Variables

Une variable représente une inconnue comme la notion traditionnelle de variable mathématique. En PC, il existe plusieurs types de domaines mais les variables à domaine fini sont les plus utilisées.

2.2.2 Domaines

Le domaine d'une variable est l'ensemble des valeurs possibles que peut prendre cette variable. En PC, les domaines peuvent être de différents types (entiers, réels ou des symboles) et peuvent représenter un ensemble non continu de valeurs. En PC, les domaines des variables sont progressivement réduits : il n'y a jamais d'ajout de valeurs. Dans toute la suite du texte, seuls les domaines à valeurs entières seront considérés.

2.2.3 Contraintes

Une contrainte est une relation entre des variables spécifiées qui impose des restrictions sur la combinaison des valeurs que ces variables peuvent prendre. Nous pouvons aussi voir les contraintes comme des relations mathématiques entre les variables de décision, par exemple $X+Y \leq 2$. Une contrainte peut être définie en *compréhension* (exemple : $Y = 2X$) ou en *extension*. La contrainte ($Y = 2X$) est définie en extension si, par exemple, les variables X et Y ne peuvent prendre que les valeurs suivantes :

$$\{(1,2),(2,4),(3,6)\}$$

Les contraintes peuvent être de type logique. Par exemple :

$$(X \geq 1)\ OU\ (3X + Y \leq 5)$$

Le nombre de variables de décision contenues dans une contrainte est appelée l'*arité* de cette contrainte. De façon plus spécifique, des contraintes contenant une, deux et trois variables différentes sont dites *unaires*, *binaires* et *ternaires*. Par exemple, la contrainte $Z \leq 1$ est unaire tandis que la contrainte $X + Y \leq 2$ est binaire.

Il est possible d'imposer des contraintes sur des contraintes. Ce nouveau type de contraintes est appelé *métacontraintes*. Par exemple, la contrainte $\sum_{i\in[1,..,30]}(X_i = 2) \leq 7$ est une métacontrainte et signifie que le nombre de fois que la contrainte $(X_i = 2)$ est satisfaite doit être inférieur ou égal à 7. Le terme $(X_i = 2)$ vaut 1 si la contrainte $X_i = 2$ est satisfaite et 0 sinon.

En PC, les contraintes sont traitées de façon individuelle, ce qui peut être un frein à l'efficacité d'un modèle de PC. Pour palier à ce défaut, des contraintes dites *globales* sont créées. Ces contraintes spéciales sont créées pour capturer des sous-structures intéressantes de certains problèmes. En PC, à chaque contrainte (simple ou globale) est associé un algorithme appelé algorithme de réduction (filtrage) de domaine. Toute contrainte globale se caractérise par l'efficacité de son algorithme spécialisé de réduction de domaine. Une contrainte globale peut être vue comme une abstraction d'un sous ensemble de contraintes. Elle se distingue aussi par sa formulation concise.

Voici une liste de quelques exemples de contraintes globales souvent utilisées :
La contrainte globale $alldifferent(X_1, X_2, .., X_n)$ *dite contrainte d'exclusion mutuelle* signifie que les valeurs des variables $X_1, X_2, ..., X_n$ doivent être toutes deux à deux différentes. Elle est équivalente en sens à l'écriture de $n\,(n-1)\,/2$ différences du type $X_i \neq X_j$ mais son algorithme de filtrage est beaucoup plus efficace.

La contrainte de cardinalité généralisée (*gcc*), aussi appelée *contrainte de distribution de valeurs*, est dénotée par $distribute(Y, V, X)$ avec $Y = (y_1, y_2, ..., y_n)$, $V = (v_1, v_2, ..., v_n)$ et $X = (x_1, x_2, ..., x_n)$ une variable quelconque (X est un vecteur de variables tandis que Y et V sont des vecteurs de constantes). La contrainte *gcc* impose

que le nombre de fois que des entrées de la variable X prennent la valeur v_i est égal à y_i. Cette contrainte *gcc* peut être représentée de la façon suivante :
$y_j = \sum_{i \in \{1,2,...,n\}} (x_i = v_j) \quad \forall\, j \in \{1, 2, ..., n\}$
Mais la formulation (*distribute*) est conseillée car elle peut fournir une réduction de domaines plus efficace. Il faut remarquer que la contrainte *gcc* est une généralisation de *alldifferent*. Pour obtenir la contrainte globale *alldifferent*, il suffit d'imposer que $y_i \leq 1 \quad \forall\, i$.
La *contrainte de relation fonctionnelle* entre deux variables, dénotée *element*(x, V, y), impose que $x = v_y$ et elle permet de lier des variables de différents types. Ici v_y est une entrée du vecteur de constantes V et son indice y est une variable.

La *contrainte de mise en boîte* ou *contrainte d'ordonnancement avec ressources cumulatives*, dénotée *cumulative*(S, P, R, k) impose que pour la quantité k de ressource disponible à tout moment, toutes les tâches doivent être planifiées de telle façon qu'on ne dépasse jamais cette capacité, avec $S = (s_1, s_2, ..., s_n)$, $P = (p_1, p_2, ..., p_n)$ et $R = (r_1, r_2, ..., r_n)$ où la tâche i débute à la période s_i et consomme r_i unités de la ressource R pendant une durée de p_i périodes.
Les contraintes globales sont utilisées aussi pour combler le fossé qui existe entre la PC et la PLNE. Nous verrons plus tard pourquoi la partie optimisation est moins forte en PC qu'en PLNE. La PC est très forte pour trouver des solutions réalisables tandis que la RO est très efficace pour résoudre des problèmes d'optimisation, ce qui motive l'idée d'intégrer la PC et la PLNE.

2.2.4 Réduction (ou filtrage) de domaine

Rappelons qu'à toute contrainte est associé un algorithme de réduction de domaine qui est utilisé pour retirer des valeurs incohérentes (par rapport à la contrainte) contenues dans les domaines des variables. Les algorithmes de réduction de domaine utilisent des techniques spécifiques appelées techniques de cohérence. Nous pouvons distinguer les techniques de cohérence de domaines. La technique de cohérence de domaine d'arité 1 assure que toutes les valeurs contenues dans les domaines des variables permettent

de vérifier les contraintes. La contrainte $X \geq 1$ avec le domaine de la variable X noté $D_X = \{2, 3, 6\}$ qui vérifie la cohérence de domaines d'arité 1.

La contrainte $X \leq Y$ vérifie la cohérence de domaines d'arité 2 car, pour toute valeur du domaine $D_X = \{1, 3, 4\}$ de X, il existe une valeur du domaine $D_Y = \{2, 4, 5\}$ de Y qui vérifie la contrainte, et *vice versa*. Cette définition de la cohérence de domaines d'arité 2 peut être étendue pour définir la notion de cohérence de domaines qui concerne les contraintes d'arité k. Soit une contrainte C contenant les variables $X_1, X_2, .., X_k$. La contrainte C est dite cohérente sur les domaines si, pour une valeur v de la variable X_i, il existe $k-1$ valeurs des autres variables $v_1, v_2, .., v_{i-1}, v_{i+1}, ..., v_k$ qui permettent de vérifier la contrainte C. Même si elles présentent l'avantage d'être indépendantes du problème traité, les techniques générales de cohérence de domaines ne sont pas performantes pour les problèmes de grande taille, ce qui motive aussi la conception et l'utilisation de contraintes globales. En effet, leur portée est limitée car, en pratique, k varie seulement de 1 à 7 et les techniques de cohérence de domaines les plus utilisées sont celles pour lesquelles k varie de 3 à 4.

Une autre alternative (moins forte cependant) à la cohérence de domaines est la cohérence de bornes. La technique de cohérence sur les bornes se concentre sur ce qui se passe sur les bornes seulement. Une contrainte est cohérente sur les bornes si, pour chaque borne du domaine de la variable considérée, il existe des valeurs des domaines des autres variables qui permettent de vérifier la contrainte. Entre d'autres termes, soit une contrainte C contenant les variables $X_1, X_2, .., X_k$. La cohérence de bornes est atteinte pour C si et seulement si :

- il existe $(v_1, v_2, .., v_{i-1}, v_i^{min}, v_{i+1}, .., v_k)$ avec v_i^{min} = valeur minimale de X_i, $v_j \in \left[v_j^{min}, v_j^{max}\right]$ et

- il existe $(v_1, v_2, .., v_{i-1}, v_i^{max}, v_{i+1}, .., v_k)$ avec v_i^{max} = valeur maximale de X_i, $v_j \in \left[v_j^{min}, v_j^{max}\right]$

Par exemple soit la contrainte $X \leq Z + 1$ avec $D_X = \{-1, 0, 1, 2, 3, 4\}$ et $D_Z = \{-3, -2, -1, 0, 1\}$. En appliquant la cohérence de bornes, les domaines de X et Z deviennent $D_X = \{-1, 0, 1, 2\}$ et $D_Z = \{-2, -1, 0, 1\}$. Les valeurs effacées du domaine d'une variable sont celles pour lesquelles il n'y a pas de valeur dans l'autre qui permettent de vérifier la contrainte.

2.2.5 Propagation de contraintes

La propagation de contraintes est un processus itératif purement déductif utilisé pour propager les conséquences des modifications des domaines des variables de décision contenues dans les contraintes. Les contraintes interagissent entre elles via les domaines des variables qu'elles ont en commun. L'algorithme de réduction de domaine associé à une contrainte est activé dès que le domaine d'une variable de cette contrainte est modifié. La modification du domaine d'une variable a lieu quand les bornes de ce domaine sont réduites, ou une valeur est supprimée, ou quand une variable a été fixée à une valeur (le domaine est réduit à un singleton). Dès que le domaine d'une variable est modifié alors chaque contrainte contenant cette variable active son algorithme de réduction de domaine et applique les techniques de cohérence. Puis une réaction en chaîne est déclenchée. À la fin du processus de propagation de contraintes, trois cas peuvent se présenter : (1) un domaine devient vide, ce qui signifie qu'il y a échec et il faut faire un retour arrière, (2) une solution est trouvée c'est-à-dire que le domaine de chaque variable est réduit à un singleton, (3) certains domaines contiennent plus d'une valeur. Dans ce cas, il est nécessaire de brancher pour explorer l'arbre de recherche.

2.2.6 Branchement et heuristiques de recherche

Si à la suite de la propagation de contraintes, il subsiste au moins une variable dont le domaine contient plus d'une valeur, alors il est nécessaire d'avoir une stratégie de

recherche pour parcourir ce qui reste de l'arbre de recherche. Soit x une variable dont le domaine contient plusieurs éléments (*i.e.* $|D_x| \geq 2$) au noeud N de l'arbre de recherche. Il faut d'abord choisir une valeur $v \in D_x$. Ensuite deux nouveaux noeuds, N_1 et N_2, sont créés. Puis la contrainte $(x = v)$ est ajoutée à l'ensemble des contraintes utilisées au noeud N pour former l'ensemble des contraintes du noeud N_1, tandis que la contrainte $(x \neq v)$ est ajoutée à l'ensemble des contraintes utilisées au noeud N pour former l'ensemble des contraintes du noeud N_2. Les contraintes $(x \neq v)$ et $(x = v)$ sont appelées *contraintes de branchement*. Toutefois, il faut mentionner qu'il est possible de fractionner les domaines des variables pour la recherche de solutions plutôt que de fixer les valeurs des variables. Par exemple, supposons qu'à un moment donné (pendant le parcours de l'arbre de recherche), la variable sélectionnée X a pour domaine $[1, .., 6]$. Le domaine de X peut être partitionné grâce aux deux contraintes dites de *branchement* $X \leq 3$ et $X > 3$. Aussi, le domaine de X pourrait être partitionné grâce à six contraintes de branchement $X = i$ avec $i = 1, .., 6$. C'est ce type de branchement (appelé *étiquetage*) qui est le plus souvent utilisé pour des variables entières.

Il apparaît clairement que dans le processus de branchement, ce sont le choix de variable, le choix de valeur et la sélection du prochain noeud à traiter (exploration de l'arbre) qui sont les points les plus importants pour la recherche de solutions d'un problème.

Parmi les heuristiques de choix de variables, on distingue :

(1) L'heuristique basée sur un ordre *statique*, c'est-à-dire que l'ordre de sélection des variables est déterminé avant le début de la recherche de solutions.

(2) L'heuristique basée sur un ordre *dynamique*, c'est-à-dire que les variables sont sélectionnées à chaque noeud en fonction du contexte courant et d'un critère donné. Ainsi, cette stratégie tire profit d'une information mise à jour. Elle est presque toujours utilisée parce qu'elle est performante (en dépit de son coût plus élevé en temps de calcul).

(3) L'heuristique *Échec en Premier* (*first-fail*). Elle consiste à sélectionner en priorité les variables les plus contraintes, c'est-à-dire à sélectionner en premier les variables qui ont le plus petit domaine.

La meilleure heuristique de sélection de variables ou de valeurs est celle qui est basée sur une analyse de l'application étudiée.

L'heuristique de choix de valeurs est moins importante que l'heuristique de choix de variable. Il est important de choisir la meilleure valeur possible aussi bien pour trouver une solution que pour la bonne qualité de la solution elle-même. Ainsi, le choix de valeurs dépend de l'application étudiée.

En ce qui concerne l'exploration de l'arbre, la stratégie de recherche en profondeur avec retour arrière chronologique est l'une des stratégies le plus souvent utilisées car elle génère peu de noeuds (consomme donc peu de mémoire).

Il existe plusieurs autres stratégies de recherche régulièrement utilisées parmi lesquelles les trois suivantes :

- la stratégie *à divergence limitée* ou *Limited Discrepancy Search* (LDS) qui est une variante de la stratégie de recherche en profondeur. On parle de *divergence* quand une branche de l'arbre de recherche suit une affectation de variable et valeur qui n'est pas la première suggérée par l'heuristique. Dans la stratégie de recherche en profondeur, une grande importance est accordée aux décisions prises tôt dans l'arbre de recherche. Alors que dans la stratégie à divergence limitée, on se permet de ne pas toujours suivre l'heuristique. Le processus de recherche commence par traverser l'arbre de recherche sans permettre aucune divergence (ce qui équivaut à toujours suivre la branche la plus à gauche comme dans la stratégie de recherche en profondeur), ensuite on tolère une seule divergence, puis deux, trois, etc. Puisque l'heuristique de recherche est supposée être bonne, il ne doit pas y avoir beaucoup de sauts dans la recherche de solutions.

- la stratégie *Slice Based Search* (SBS) qui consiste aussi à limiter le nombre de décisions qui vont à l'encontre de l'heuristique (encore supposée bonne). C'est une variante de la stratégie LDS. Dans la stratégie SBS, les divergences varient dans un intervalle plutôt que d'être régulièrement incrémentées d'une unité. La motivation derrière l'utilisation de cette stratégie est que lorsqu'il y a un échec, cela peut vouloir dire que l'on peut trouver une solution si des décisions légèrement différentes sont prises (donc il faut changer un peu les décisions prises par l'heuristique). Plus il y a des échecs dans la recherche de solutions réalisables, moins on fait confiance à l'heuristique.

- la stratégie *Depth-Bounded Discrepancy Search* (DDS) est une variante de la stratégie SBS. Elle favorise des divergences tôt et haut dans l'arbre de recherche. L'idée derrière cette stratégie consiste à considérer que l'heuristique de recherche aura tendance à échouer tôt à cause d'un manque d'informations. À chaque itération, DDS essaie le nombre de décalages permis avant de suivre l'heuristique de recherche.

2.2.7 Optimisation

L'optimisation en PC fonctionne de la façon suivante : pour un problème de minimisation, la première étape consiste à résoudre d'abord un PSC. Puis à chaque fois qu'une solution réalisable $x^\star$ est trouvée, la fonction objectif $g(x^\star)$ et la contrainte $g(x) < g(x^\star)$ est ajoutée. Ensuite, un PSC est à nouveau résolu avec le nouvel ensemble de contraintes. Le processus s'arrête quand il n'y a pas de solution au PSC résolu, la valeur trouvée à l'étape précédente est alors prise comme solution optimale du problème d'optimisation.

Cette méthode qui consiste à transformer un problème d'optimisation en une séquence de PSC est faible. Aucune autre information n'est utilisée pour améliorer l'objectif. C'est ce qui motive la conception de méthodes hybrides pour tirer profit de façon complémentaire des forces de la PC et de la PLNE.

2.3. Méthodes hybrides de la programmation par contraintes et la recherche opérationnelle

Une méthode hybride de la PC et de la RO peut être vue comme une méthode utilisant des techniques classiques de ces deux communautés pour atteindre une plus grande efficacité. Dans cette section, les cadres d'unification de la PC et de la RO sont d'abord présentés, ensuite les méthodes hybrides suivantes sont décrites : les contraintes globales, les méthodes hybrides basées sur la coopération, les principales méthodes hybrides de décomposition et les méthodes de recherche locale combinées avec la PC.

Cadres d'unification de la programmation par contraintes et la recherche opérationnelle

Parmi les travaux les plus significatifs sur l'intégration de la PC et de la recherche opérationnelle, il faut mentionner ceux de Bockmayr et Kasper (1998), Hooker (2003), Thorsteinsson (2001b) et Hooker (2005). Poursuivant la logique d'unification de la PC et la PLNE, Hooker (2005) propose un cadre algorithmique commun dénommé *Search-Infer-and-Relax*. Cette structure commune à la PC et à la PLNE est étendue à d'autres méthodes d'optimisation combinatoire comme par exemple la décomposition de Benders, les méthodes basées sur les coupes (*nogoods*) et diverses métaheuristiques. Hooker (2005) présente une vue globale de ce cadre mais aussi leur fonctionnement et interaction dans plusieurs méthodes d'optimisation combinatoire. Son objectif est de bâtir une théorie unificatrice des diverses méthodes de résolution de problèmes d'optimisation combinatoire pour permettre par la suite la création d'un solveur unique comme plateforme d'utilisation et d'intégration.

Contraintes globales

À l'origine de la PC, les contraintes sont traitées de façon séquentielle, l'une après l'autre, mais ce point de vue local a vite montré ses limites. Ainsi est venue l'idée

de traiter en bloc certains sous-ensembles de contraintes pour améliorer la réduction de domaines et la recherche de solutions. Ces blocs de contraintes correspondent souvent à des sous-structures qui peuvent être résolues efficacement (par les techniques de la recherche opérationnelle par exemple). Ces blocs de contraintes sont communément appelés contraintes globales et peuvent être vues comme une conjonction de contraintes ou une relaxation d'un problème plus grand. Le lecteur est referé à Milano (2004) pour une description détaillée des contraintes globales les plus souvent utilisées dans la littérature.

Régin (1996, 1997, 1999a, 1999b, 2002, 2005) est l'un des pionniers dans le domaine de la conception de contraintes globales. D'autres auteurs comme Zhou (1996, 1997), Melhorn et Thiel (2000) et Pesant (2001, 2004) ont conçu des contraintes globales intéressantes. La tendance actuelle consiste à concevoir des contraintes globales qui prennent en compte le coût. L'idée principale est d'effacer du domaine des variables les combinaisons de valeurs qui n'améliorent pas le coût total. Parmi les premiers travaux faisant référence à ce type de contraintes globales on retrouve celui de Focacci *et al.* (1999a, 1999b). Leur algorithme de filtrage de domaines est basé sur les coûts réduits. Ils ont intégré une composante optimisation dans une contrainte globale. Dans Régin (2002), on retrouve ce type de containtes globales. Une autre tendance actuelle dans la conception de contraintes globales consiste à permettre la violation de certaines contraintes globales puis d'y associer un coût (voir Beldiceanu et Petit (2004), Van Hoeve *et al.* (2005)).

Méthodes hybrides basées sur la coopération

Les méthodes hybrides basées sur la coopération sont des méthodes conçues pour résoudre le même problème avec deux solveurs différents (PC et PLNE). Dans ce type de méthodes, le problème maître est souvent modélisé avec la PC. Une idée importante souvent rencontrée est l'utilisation de la solution optimale du sous-problème linéaire pour guider la recherche de solutions au modèle original. Un des premiers travaux significatifs dans le domaine des méthodes hybrides basées sur la coopération est celui

de Rodosek *et al.* (1999). Ils ont conçu une procédure systématique de linéarisation de contraintes globales pour créer un modèle *caché* en nombres entiers équivalent à celui formulé avec la PC. Leurs travaux font suite à ceux de Hajian *et al.* (1995) sur ce thème. Les travaux de Thorsteinsson et Ottosson (2002) intégrent deux directions de recherche. La première est la linéarisation automatique de contraintes disjonctives ou non linéaires, ce qui entraîne l'introduction de variables auxiliaires comme dans Refalo (2000) et Rodosek *et al.* (1999). La seconde est l'utilisation de coûts réduits d'un sous-problème d'affectation pour la propagation (voir Rodosek *et al.* (1999)).

El Sakkout et Wallace (2000) ont proposé une méthode de décomposition originale pour résoudre une variante dynamique d'un problème d'ordonnancement classique. En s'appuyant sur un plan d'ordonnancement établi, leur objectif consiste à minimiser la somme des déviations entre les temps de début réels des activités et les temps de début établis. Par la suite, Beck et Refalo (2003) se sont inspirés de ces travaux pour concevoir un modèle hybride de décomposition qui minimise les coûts d'avance et de retard des tâches de production.

Refalo (1999) a présenté une approche qui permet d'ajouter des coupes à la formulation linéaire du problème au cours de la propagation de contraintes. Une caractéristique de son approche est qu'il travaille avec l'enveloppe convexe d'une fonction linéaire par morceaux contenue dans une contrainte.
Bockmayr et Kasper (1998) ont proposé un cadre global d'intégration de la PC et la PLNE. Ils s'appuient sur la similarité de concepts utilisés dans les deux domaines (la PC et la PLNE servent toutes les deux à résoudre des problèmes complexes d'optimisation combinatoire mais l'inférence et la recherche de solutions y sont mises en oeuvre différemment). Hooker et Osorio (1999) ont par la suite proposé le cadre dénommé Mixed Logical/Linear Programming. Dans ce cadre, l'utilisateur peut séparer les variables en deux parties : une partie constituée de variables discrètes et une autre formée de variables continues.

$$\begin{aligned} & Min \;\; c^T x \\ sous\ les\ contraintes \;\; : \;\; & h_i(y) \rightarrow A^i x \geq b^i \quad \forall\, i \in I \\ & x \in R^n, y \in D \end{aligned}$$

L'idée principale consiste à satisfaire les contraintes $h_i(y)$ contenant les variables discrètes avec la PC. Ensuite, une fois que des déductions sont tirées de la solution précédente, la relaxation (contenant les contraintes à variables continues $A^i x \geq b^i$) du modèle original est résolue. Cette relaxation est typiquement un programme linéaire de petite taille à structure spéciale qui peut être efficacement résolu.

L'exploitation de la relaxation linéaire d'un problème pour accroître l'efficacité de la PC est une idée souvent retrouvée dans la conception de méthodes hybrides, parce que la partie optimisation de problèmes est le point faible de la PC (le lien entre la fonction objectif et les variables de décision est plutôt faible). Ainsi, la relaxation linéaire est souvent utilisée pour guider la recherche de solutions (voir El Sakkout et Wallace (2000) et Ajili et El Sakkout (2003)). Pour conclure cette sous-section, nous mentionnerons l'existence de ECLiPSe (voir Group (2002)) qui est une plateforme de développement de méthodes hybrides de décomposition. La proposition de cadres théoriques d'intégration de la PC et la PLNE nécessite l'existence de telles plateformes. Outre la programmation logique par contraintes, ECLiPSe possède une interface commune aux solveurs commerciaux (CPLEX et XPRESS-MP (2001)) et supporte les techniques de programmation mathématique et programmation stochastique. ILOG CONCERT (2003) et MOSEL (2004) constituent aussi des plateformes d'intégration de la PC et de la PLNE.

Méthodes hybrides de décomposition

Les méthodes hybrides de décomposition sont des méthodes conçues pour résoudre différents problèmes avec plusieurs solveurs. Un grand nombre de méthodes de décomposition hybrides utilise la PC pour résoudre le problème maître du fait de sa capacité à s'adapter aux variations du problème en main (changements dans la taille du problème, la structure de données ou ajout de contraintes additionnelles). L'idéal est d'avoir ensuite un sous-problème linéaire à résoudre. Toutefois, il faut remarquer que le schéma de décomposition dépend fortement du problème étudié et les travaux

qui portent sur l'unification des deux approches se basent sur une analyse des similarités des concepts utilisés.

Les méthodes hybrides de décomposition peuvent être classées en trois grands groupes (toutefois, il existe des méthodes de décomposition qui ne sont pas insérées dans ces groupes) : (1) génération de colonnes et programmation par contraintes ; (2) décomposition de Benders et programmation par contraintes ; (3) décomposition lagrangienne et programmation par contraintes.

- **Génération de colonnes et programmation par contraintes**

  La plupart des travaux présentés ont pour but de prouver la pertinence de l'intégration des techniques de la PC et de la recherche opérationnelle par le gain potentiel d'efficacité. Dans la plupart des travaux dans ce domaine, le sous-problème est modélisé et résolu avec la PC alors que dans la méthode de génération de colonnes classique, le sous-problème est souvent modélisé et résolu avec la programmation dynamique.

  Dans les travaux sur les méthodes de génération de colonnes basée sur la PC, on retrouve souvent l'idée de résoudre un problème de satisfaction de contraintes (PSC) permettant de traiter des règles complexes qui rendent le sous-problème difficile à résoudre. La méthode de génération de colonnes classique est très efficace pour résoudre des problèmes de très grande taille où les sous-problèmes ont une structure particulière (le sous-problème est présenté sous forme d'un problème de plus court chemin avec contraintes de ressources). Par contre, lorsque le sous-problème est difficile (règles de travail et autres contraintes pratiques), la modélisation avec la méthode classique de génération de colonnes devient plus complexe.

  La flexibilité de la PC permet de prendre en compte ce type de contraintes. Dans les problèmes de génération de colonnes, le sous-problème peut être vu comme un problème de réalisabilité (*i.e* problème de satisfaction de contraintes -PSC-) plutôt qu'un problème d'optimisation. Comme la programmation dynamique, la PC

permet de générer plusieurs colonnes à la fois. La PC offre plus de flexibilité pour concevoir des stratégies de recherche spécialisées au problème étudié. Toutefois, tout comme la méthode de génération de colonnes classique, la méthode de génération de colonnes basée sur la PC est aussi confrontée à des problèmes de stabilisation. Les problèmes de convergence peuvent même être plus aigus dans la méthode de génération de colonnes basée sur la PC (voir Rousseau (2004)). Gendron *et al.* (2005) ont présenté et comparé deux stratégies de recherche conçues pour résoudre le sous-problème avec la PC. Ces deux stratégies servent à générer des colonnes plus prometteuses. Les tests ont été faits sur un problème d'ordonnancement de tâches de médecins.

La méthode de génération de colonnes qui utilise la PC pour résoudre le sous-problème est efficace dans certaines classes du problème d'affectation généralisé (voir Savelsberg (1997)), dans les problèmes de tournées d'équipes sportives (*Traveling Tournament Problem*, voir Easton *et al.* (2002)) et les problèmes des golfeurs de loisir (*Social Golfer Problems*, voir CSPLib). Le problème de tournées d'équipes sportives consiste à organiser un tournoi d'équipes sportives dans lequel l'objectif consiste à minimiser la distance totale parcourue par les équipes. Il y a des rencontres *aller et retour* et chaque équipe commence à domicile et y retourne à la fin du tournoi. Pour chaque équipe, trois rencontres consécutives ne peuvent avoir lieu à domicile (il en est de même pour les rencontres à l'extérieur). En outre, deux rencontres *aller et retour* ne doivent pas se suivre. Le problème des golfeurs de loisir consiste à programmer les rencontres de g groupes de k joueurs de golf chaque semaine pendant s semaines de telle sorte que deux joueurs de golf ne peuvent se retrouver au sein d'un même groupe plus d'une fois. La formulation du problème des golfeurs de loisir (qui est en fait un PSC) en un problème de génération de colonnes basée sur la PC permet d'éviter les symétries. Dans les travaux de Savelsberg (1997) et Easton *et al.* (2002), la formulation compacte du problème permet de concevoir une stratégie de branchement efficace.

Des travaux sur la méthode de génération de colonnes basée sur la PC ont été réalisés dans le domaine du transport aérien (problèmes d'affectation d'équipages)

et dans les problèmes de tournées de véhicules. Junker *et al.* (1999) ont proposé un cadre de décomposition qui utilise la PC pour résoudre le sous-problème qui est formulé sous forme de PSC. Le cadre proposé permet de résoudre efficacement des problèmes d'ordonnancement quand le graphe associé est naturellement acyclique. À la suite de Junker *et al.* (1999), Fahle *et al.* (2000b), ont résolu le sous-problème sous forme de PSC et il consiste à trouver des équipages qui respectent les règles de travail avec un coût réduit négatif. Ils ont aussi proposé une contrainte globale définie pour les graphes orientés acycliques qui est basée sur une condition sur les coûts réduits négatifs. Par la suite, Sellmann *et al.* (2000) ont développé un algorithme hybride qui s'appuie sur la combinaison d'une méthode de génération de colonnes basée sur la PC et d'une heuristique de recherche basée sur la PC pour résoudre un problème d'affectation d'équipages dans une compagnie aérienne européenne. Leurs efforts sont concentrés sur la génération de colonnes de bonne qualité pour améliorer la fonction objectif.

Une autre classe de sous-problèmes souvent rencontrée dans la méthode de génération de colonnes classique sont les sous problèmes de type sac-à-dos (*Knapsack Problems*). Ce type de sous-problèmes est rencontré dans les problèmes de découpe (*Cutting Stock Problems*). Des contraintes pratiques (qui peuvent être non linéaires) gâchent la structure du sous-problème lorsqu'elles sont ajoutées aux contraintes sur les patrons de découpe, ce qui est à l'origine de l'approche de Fahle et Sellmann (2000a). Ils ont développé une méthode de génération de colonnes basée sur la PC où la PC résout le sous-problème de sac-à-dos qui contient des contraintes additionnelles. Ils ont obtenus des résultats intéressants mais comme ils l'ont mentionné, les outils qu'ils ont développés ne sont pas comparables aux meilleurs outils utilisés pour les problèmes de sac-à-dos *classiques*.

Pour modéliser le sous-problème dans le contexte d'un problème de tournées de véhicules de petite taille (où le graphe associé a une structure cyclique), Rousseau *et al.* (2002b) ont proposé un algorithme de plus court chemin basé sur la PC et la programmation dynamique. Les problèmes considérés dans leurs travaux sont dits problèmes de tournées profitables avec contraintes de ressources qui sont essentiellement des problèmes de plus court chemin cyclique avec contraintes de ressources.

Demassey *et al.* (2005) ont utilisé une méthode de génération de colonnes basée sur la PC pour résoudre un problème d'ordonnancement de tâches dans le contexte du commerce de détail. La PC est utilisée pour résoudre le sous-problème et elle permet de générer des quarts de travail de moindre coût grâce à une nouvelle contrainte globale (*cost-regular*). Cette contrainte globale est une variante de la contrainte globale *regular* (Pesant (2001b)). Leurs résultats sont préliminaires et cette approche a besoin d'être améliorée pour être en mesure de travailler sur des données réelles et pour une prise en compte d'hypothèses plus réalistes (tous les employés ne sont pas interchangeables). Bien qu'ils aient utilisé des données de problèmes de sac-à-dos *classiques*, ils en ont omis certaines classes.

- **Décomposition de Benders et programmation par contraintes**

Dans la décomposition de Benders classique, aussi bien le problème maître que les sous-problèmes sont souvent résolus par la programmation linéaire. Cependant, dans les méthodes hybrides qui utilisent la décomposition de Benders, le problème maître ou les sous-problèmes peuvent être modélisés et résolus avec la PC. Eremin et Wallace (2001) ont présenté avec ECLiPSe un modèle dans lequel le problème maître est résolu avec la PC tandis que les sous-problèmes sont modélisés en programmation linéaire. L'intérêt principal de leur implantation est que le programmeur a seulement besoin de spécifier quelles variables appartiennent à quels sous-problèmes. Par la suite, la forme duale de chaque sous-problème est extraite puis la décomposition de Benders est identifiée, ce qui facilite la tâche aux chercheurs qui sont spécialisés dans un domaine unique (recherche opérationnelle ou PC) pour le développement d'algorithmes hybrides. Benoist *et al.* (2002) ont présenté une approche similaire à la précédente dont les avantages sont explicites. Comme dans l'approche de Eremin et Wallace (2001), Benoist *et al.* (2002) ont modélisé le problème maître avec la PC tandis que le sous-problème est résolu par la programmation linéaire. Le problème maître contient une contrainte (appelée *flow*) qui utilise la sous-structure de flux du modèle(conservation du flux, flux borné supérieurement et inférieurement). La contrainte globale *flow* s'inspire de la résolution

du problème du flux maximum. Cette méthode de décomposition a été appliquée à un problème de confection d'horaires dans une centrale d'appels téléphoniques. Aussi, elle s'est montrée supérieure à la méthode classique (formulée avec la PLNE) lorsque les contraintes deviennent complexes (par exemple les domaines des heures de travail contiennent des trous).

D'autres auteurs ont modélisé le problème maître avec la PLNE tandis que les sous-problèmes sont résolus avec la PC. Hooker (2000), Jain et Grossmann (2001) et Thorsteinsson (2001) utilisent ce type de décomposition. De façon globale, la PC leur permet de générer des coupes tandis que le problème maître leur permet de prendre des décisions basées sur le coût (affectation de tâches sur les machines par exemple). Les sous-problèmes (un par ressource) vérifient la réalisabilité des affectations et génèrent des coupes en cas de contradiction. Thorsteinsson (2001) présente un cadre de décomposition qui inclut la décomposition de Benders. Jain et Grossmann (2001) présentent une méthode de décomposition où le problème maître est modélisé avec la PLNE tandis que les sous-problèmes (de satisfaction de contraintes) sont résolus avec la PC. Ils ont appliqué leur approche à un problème d'ordonnancement de machines parallèles différentes. Ils proposent aussi un algorithme de *Branch-and-Bound* qui est basé sur une combinaison de la programmation linéaire et la PC. À la suite de ces travaux, Hooker et Ottosson (2003) ont introduit une généralisation de la décomposition de Benders basée sur la logique étendue à la PLNE. Cette approche semble plus efficace que la décomposition de Benders classique seulement quand le nombre de sous-problèmes n'est pas grand. Toutefois, les coupes en nombres entiers sont obtenues après la linéarisation de coupes contenant beaucoup de disjonctions. Une telle linéarisation crée beaucoup de variables auxiliaires, ce qui complique davantage le problème maître. Avec cette approche hybride, Hooker et Ottosson (2003) ont résolu une classe de problèmes de planification et d'ordonnancement.

Chu et Xia (2004) ont proposé une nouvelle méthode pour générer des coupes. Ces coupes proviennent d'autres coupes exprimées sous forme disjonctive. Toutefois, les coupes de Benders en nombres entiers ne sont pas toujours disponibles

car des conditions de qualification s'appliquent à la forme disjonctive précédente. Maravelias et Grossmann (2004) ont modélisé leur problème maître avec la PLNE tandis que les sous-problèmes sont des problèmes de satisfaction de contraintes résolus avec la PC. Au cas où il n'y a pas de solution réalisable, une coupe en nombres entiers est générée. Leur approche a été appliquée pour un problème d'ordonnancement dans le contexte d'une industrie chimique.

- **Décomposition lagrangienne et programmation par contraintes**

Benoist *et al.* (2001) ont proposé une méthode hybride pour résoudre un problème de tournées d'équipes sportives. Leur approche consiste à dualiser les contraintes de liaison (*coupling constraints*) puis de résoudre une succession de problèmes indépendants avec la PC (plus précisément un solveur du problème du voyageur de commerce basé sur la PC). La borne inférieure lagrangienne est obtenue grâce à une contrainte globale contenue dans le problème maître. Sellmann et Fahle (2003) ont présenté une méthode de relaxation lagrangienne basée sur la PC pour résoudre un problème d'enregistrement automatique en télévision qui contient une contrainte de type sac-à-dos et une contrainte de type ensemble stable de taille maximale dans un graphe intervalle. Leur approche est décrite de la façon suivante : étant donné un problème d'optimisation dans lequel l'ensemble des contraintes peut être divisé en deux parties A et B, pour chacun de ces deux sous-ensembles, il existe un algorithme efficace de propagation de contraintes mais il n'existe pas d'algorithmes de propagation de contraintes efficace pour résoudre simultanément les deux familles de contraintes. La famille de contraintes A (respectivement B) est dualisée puis l'algorithme efficace de propagation de contraintes de B (respectivement A) est appliqué. L'approche de Sellmann et Fahle (2003) a donné des résultats intéressants et elle peut être généralisée à plus que deux sous-ensembles de contraintes. Dans les deux précédents articles, les bornes de la fonction objectif du modèle original permettent la réduction du domaine des variables.

Recherche locale et programmation par contraintes

À quelques exceptions près, les méthodes de recherche locale sont souvent plus efficaces que les méthodes exactes pour résoudre des problèmes pratiques de très grande taille (les problèmes de tournées de véhicules par exemple). Toutefois, leur manque de flexibilité constitue leur principal défaut. En effet, la mise à jour ou l'ajout de contraintes peut sensiblement dégrader les performances d'une méthode de recherche locale qui donnait de très bonnes solutions avant modifications. C'est la volonté de gagner plus de flexibilité qui motive les travaux d'hybridation de la PC et des méthodes de recherche locale. La flexibilité dans la mise à jour de contraintes ou la prise en compte de contraintes additionnelles est l'un des points forts de la PC. Aussi, il est naturel de penser à l'hybridation de la PC et des méthodes de recherche locale car la PC utilise beaucoup d'heuristiques dans les stratégies de recherche. Dans les méthodes hybrides qui combinent la PC et les méthodes de recherche locale, la PC est souvent utilisée sous forme de sous-problème. Toutefois, il existe des travaux où les techniques de recherche locale sont utilisées à l'intérieur de la PC. C'est le cas des travaux de De Backer *et al.* (2000) qui ont utilisé la recherche locale et les métaheuristiques dans la stratégie de recherche dans la PC pour résoudre des problèmes de tournées de véhicules. La PC y est utilisée seulement pour vérifier la validité des solutions et fixer la valeur des variables de décision. Pour mieux guider la stratégie de recherche de solutions, Caseau et Laburthe (1999) présentent une heuristique pour résoudre des problèmes de tournées de véhicules de grande taille. Leur approche est basée sur une version incrémentale d'une stratégie de recherche de solutions (la stratégie d'insertion *Look-ahead*) combinée à une procédure d'optimisation.

Caseau *et al.* (2001) présentent des pistes pour l'hybridation des méthodes de recherche locale et la PC. Ils utilisent les techniques de recherche globale pour explorer intensivement les voisinages. Ils montrent comment des structures spécialisées peuvent favoriser la conception et l'insertion dans la PC d'heuristiques basées sur les méthodes de recherche locale et globale. Ils ont fait leurs tests sur des problèmes de tournées de véhicules et des problèmes d'ordonnancement. D'autres auteurs comme Pesant et Gendreau (1996 et 1999), Pesant *et al.* (1997) Rousseau *et al.* (2002) se sont intéressés à concevoir de meilleurs voisinages. Ce sont surtout des voisinages de grande

taille conçus pour accroître la probabilité de trouver une meilleure solution plus vite et pour éviter les optima locaux. Il faut mentionner que Pesant et Gendreau (1996, 1999) ont proposé un cadre général d'intégration de méthodes de recherche locale et de la PC. Par la suite, Rousseau *et al.* (2002a) ont proposé une méthode hybride qui combine la PC, la recherche locale et des idées provenant des algorithmes génétiques pour résoudre un problème de confection d'horaires de médecins dans deux hopitaux. Il faudrait aussi mentionner les travaux de Michel et Van Hentenryck (2000, 2004) dans la conception de languages de modélisation qui intégrent la PC et les méthodes de recherche locale.

Les trois méthodes présentées dans cette thèse sont des méthodes de décomposition hybrides car nous avons à chaque fois un problème maître et un sous-problème résolus par deux solveurs différents (PC et PLNE). En particulier notre deuxième méthode de décomposition est inspirée de la décomposition de Benders.

CHAPITRE 3 : SYNTHÈSE DES TRAVAUX

Dans cette thèse, deux problèmes similaires dans le contexte d'un atelier flexible et un troisième dans le contexte d'une mine souterraine sont étudiés. Ces trois problèmes sont des problèmes intégrés d'ordonnancement et routage sans conflits de chariots. Dans chacun de ces problèmes, une méthode de décomposition hybride qui combine la programmation par contraintes (PC) et la programmation linéaire en nombres entiers (PLNE) est proposée. Dans chaque méthode de décomposition, le problème maître (ordonnancement) est modélisé avec la PC tandis que le sous-problème (routage sans conflit) est modélisé avec la PLNE.

La première méthode de décomposition est à la base des deux algorithmes développés suivants. L'existence d'un très grand nombre de solutions optimales pour le modèle de PC mais irréalisables pour le modèle de PLNE a conduit à l'exploration d'une autre approche dans la seconde méthode de décomposition avec des améliorations dans les modèles de PC, de PLNE et l'introduction de coupes logiques de réalisabilité. La conception et l'implantation de cette méthode a donné de bons résultats dans le contexte de l'atelier flexible. Mis à part les problèmes à deux chariots, le modèle de PC a donné des résultats très satisfaisants. Il faut mentionner que le modèle de PC fait aussi l'ordonnancement implicite des tâches aux machines situées aux points de livraison. Le modèle de PLNE assure un routage exact des chariots. Du point de vue de la PC, le problème du routage exact des chariots est modélisé de façon contre intuitive car il s'agit d'un PSC. La difficulté à concevoir une stratégie de recherche pour ce PSC nous a amené à concevoir un modèle de PLNE pour le routage exact des chariots. L'ajout graduel de chariots permet souvent d'abaisser les niveaux de retards de production mais engendre un peu plus de congestion. En conséquence, la taille du modèle de PLNE augmente et ceci rallonge le temps de résolution des problèmes traités.

En ce qui concerne l'atelier flexible, le choix doit être fait entre résoudre la totalité des problèmes (problèmes avec trois chariots) et minimiser systématiquement les

retards de production en utilisant le plus de chariots au risque de dépasser légèrement la limite de temps de résolution (fixée arbitrairement). La majeure partie du temps de résolution est consommée par le modèle de PLNE, ce qui est aisément compréhensible. À cause de la grande taille du modèle de PLNE (graphe espace - temps), le temps de réponse négative peut être long dans le cas où la relaxation linéaire du modèle de PLNE a une solution sans qu'une solution entière ne soit disponible. L'utilisation de coupes logiques de réalisabilité permet de chercher les solutions alternatives résiduelles. Toutefois, leur portée est limitée car elles sont très disjonctives et leur ajout affaiblit graduellement le modèle de PC à chaque itération. Une disjonction est équivalente à deux noeuds de branchement créés dans l'arbre de recherche du modèle de PC. Ainsi, trois directions de recherche sont mises en évidence :

- améliorer les modèles de PC et PLNE.

- supprimer les coupes logiques. Se placer dans une logique d'anticipation et prévention de conflits plutôt que de les réparer après leur apparition.

- choisir un contexte où le problème de congestion se pose avec plus d'acuité pour mieux étudier la notion de congestion.

Ce dernier point est traité dans la troisième méthode de décomposition qui porte sur une application dans une mine souterraine. En outre, le traitement exact de l'orientation des chariots est fait puisque dans cette application, les chariots se présentent aux points de chargement et déchargement avec la pelle vers l'avant. Le contexte de la mine souterraine permet d'étudier plus en profondeur la notion de congestion. Avant d'aller plus loin dans l'analyse des résultats obtenus dans ce contexte, il convient de faire l'inventaire des caractéristiques des deux dernières méthodes de décomposition :

- dans ces algorithmes, l'approche utilisée est exacte. La méthode décompose le problème en deux sous-problèmes : une partie ordonnancement résolue par la PC tandis que la partie routage sans conflits est résolue par la PLNE.

- dans les contextes de l'atelier flexible et la mine souterraine, les problèmes résolus contiennent jusqu'à six chariots. Les caractéristiques de ces contextes ne justifient pas l'utilisation d'un plus grand nombre de chariots.

- dans le contexte de l'atelier flexible, l'horizon est fixé à 150 périodes, la limite de temps de résolution est douze minutes. 75 problèmes contenant jusqu'à treize requêtes de transport sont considérés. Une requête est équivalente à deux tâches (une collecte et une livraison). Dans le contexte de la mine souterraine, 115 problèmes sont considérés, avec jusqu'à quinze requêtes. Il n'y a pas d'horizon fixé et la limite de temps de résolution est 20 minutes.

- dans le contexte de l'atelier flexible, il y a des bornes inférieures sur les temps de début des tâches. Les contraintes de priorité sont connues *a priori*. L'ordonnancement implicite des tâches de traitement est fait sur les machines situées sur les points de livraison tandis que, dans le contexte de la mine souterraine, les contraintes de priorité sont induites après analyse du problème. Il n'y a pas de bornes sur les temps de début des tâches.
- dans le contexte de l'atelier flexible, le logiciel Ilog OPLStudio 3.6 (Ilog Solver 5.2 pour la PC, CPLEX 8.0 pour la PLNE) a été utilisé sur un ordinateur Pentium 4 (1.5 Ghz et 512 Mo de RAM). Dans le contexte de la mine souterraine, Ilog OPLStudio 3.7 (Ilog Scheduler 6.0 pour la PC, CPLEX 9.0 pour la PLNE) a été utilisé sur un ordinateur Pentium 4 (2.4 GHz, 512 Mo de RAM).

- dans le contexte de l'atelier flexible, le réseau physique est généré à chaque itération (ce qui coûte quelques dixièmes de secondes à chaque fois) tandis que, dans le contexte de la mine souterraine, le réseau physique est généré une seule fois en amont.

- dans le contexte de l'atelier flexible, l'algorithme est conçu pour un réseau quelconque, tandis que, dans le contexte de la mine souterraine, l'algorithme est conçu pour un réseau spécifique (en arbre). La tendance actuelle consiste à adapter chaque algorithme à la topologie du réseau vu le niveau de difficulté des problèmes traités.

- l'article sur la mine souterraine doit être considéré comme une amélioration de la deuxième méthode de décomposition avec une meilleure prise en compte de la congestion. Enfin l'article sur la mine souterraine se démarque par le traitement exact de l'orientation des chariots, la prévention des conflits, l'analyse de l'évolution de la mine et de son potentiel (possibilité de réaffectation rapide des chariots suite à une panne).

Les résultats obtenus dans le contexte de la mine souterraine sont tout à fait intéressants. Le modèle de PC est très efficace car sa conception est basée sur l'analyse de la topologie du réseau. À l'opposé de la borne supérieure, la borne inférieure est de très bonne qualité et c'est ce qui favorise la convergence rapide vers la solution optimale dans le modèle de PC. En outre, les contraintes de priorité induites et la notion de gel temporaire des galeries (pour l'exécution des tâches) ont un effet bénéfique sur la qualité de la solution fournie. Pour ce dernier point, les résultats obtenus montrent que très souvent, un traitement préventif presque complet des conflits est fait dans les galeries de tâches (galeries dont l'une des extrêmités est un point de chargement ou déchargement). Les parties du réseau où les conflits résiduels peuvent apparaître sont identifiables. Ce sont en très grande partie les galeries intermédiaires. Des solutions sont proposées pour remédier à ces difficultés.

La principale force du modèle de PLNE réside dans sa capacité à fournir une orientation cohérente des chariots dans un réseau très contraint. Les solutions obtenues évitent aussi beaucoup de déplacements inutiles pour des raisons d'économie d'énergie ou de sécurité.

De façon globale, l'algorithme conçu dans le contexte de la mine souterraine contient deux autres qualités. D'une part, il permet d'avoir une bonne idée de l'évolution de la mine, ce qui permet de décider en amont du dimensionnement de la flotte de chariots. D'autre part, notre méthode permet d'analyser des scénarios de pannes dans la mine avec le gel partiel ou total d'une ou plusieurs galeries. Le contexte étudié est, en effet, sujet à des pannes fréquentes (bris de chariots, chutes de roches ou fuites

d'eau). Ainsi, il est intéressant d'avoir une idée précise du fonctionnement de la mine lors de l'apparition de pannes avec certaines galeries inutilisables. Il est ainsi possible de décider du nombre convenable de chariots à utiliser pour assurer une certaine fluidité des mouvements dans la mine souterraine. Toutefois, la grande taille et le temps de réponse du modèle de PLNE constituent des limites à notre approche. Dans les contextes étudiés, nous avons besoin de ce type de graphe car il faut connaître la position de chaque chariot à tout moment. En ce qui concerne le temps de réponse qui peut parfois être long, certains modèles de grande taille de PLNE peuvent posséder une relaxation linéaire avec solution sans qu'une solution entière ne soit disponible. Les résultats obtenus sont en partie surprenants car, dans certaines instances, l'ajout d'un chariot ne fait pas baisser la durée totale des tâches. Ce phénomène s'explique par le fait que l'ajout d'un chariot peut augmenter la congestion. Aussi son orientation à son point de départ détermine si un changement d'orientation est nécessaire avant d'aller exécuter une première tâche de chargement de minerai. La proximité de son point de départ au premier point de tâche a aussi un impact sur la durée totale de toutes les tâches.

Les principaux objectifs visés et atteints au cours de nos travaux sont les suivants :

- intégrer deux niveaux d'optimisation (ordonnancement et routage sans con-flits) dans des problèmes complexes de logistique.

- explorer les méthodes de décomposition hybrides de la PC et la PLNE avec comme objectif l'intégration de leurs forces. Nous voulons profiter de l'efficacité de la PC pour traiter la partie ordonnancement et modéliser les contraintes logiques tout en gardant ce qui fait la force de la PLNE dans la partie routage sans conflits.

- concevoir des méthodes fiables qui prennent en compte des contraintes réalistes et qui traitent des aspects souvent négligés dans la littérature comme l'orientation des chariots. Les scénarios de bris de véhicules et d'accidents sont analysés et nous avons donné des réponses sur la façon efficace de réagir suite à ces événements.

- résoudre par une méthode exacte des problèmes intégrés d'ordonnancement et de routage sans conflits contenant jusqu'à six chariots. Les meilleurs résultats auparavant obtenus par une approche exacte concernaient des problèmes contenant jusqu'à quatre chariots.

Les trois algorithmes contenus dans cette thèse montre que notre méthode de décomposition hybride consiste d'abord à résoudre une relaxation non linéaire du modèle original par la PC puis à se brancher simultanément sur toutes les contraintes de routage sans conflits grâce à la PLNE. La relaxation consiste en l'affectation ordonnée des requêtes aux chariots. Dans cette relaxation, les plus courts chemins entre points de tâches sont utilisés pour approximer les routes des chariots. Le choix de la relaxation est motivé par la nécessité de savoir quelles requêtes de transport sont affectées à quel chariot et à quels moments les tâches doivent débuter. La PC est choisie pour résoudre la relaxation à cause de la présence de contraintes non linéaires mais aussi parce qu'elle est très efficace pour résoudre des problèmes d'ordonnancement. La solution de la relaxation est de bonne qualité car : (1) elle est souvent proche de la solution optimale du modèle original ; (2) le nombre d'itérations nécessaires pour atteindre la solution optimale du modèle original n'est pas grand.

La partie relaxée est l'ensemble des contraintes de routage sans conflits. Le branchement est fait simultanément sur l'ensemble de ces contraintes car le choix séquentiel de routes de chariots n'est pas une bonne stratégie. Pour obtenir cette vue globale du routage sans conflits, notre choix s'est porté sur la PLNE comme outil. La PC n'est pas utilisée dans cette partie car l'absence de vue globale constitue son principal défaut (les contraintes sont traitées de façon séquentielle).

CHAPITRE 4 : DEUX PROBLÈMES INTÉGRÉS D'ORDONNANCEMENT ET DE ROUTAGE SANS CONFLIT DANS UN ATELIER FLEXIBLE

4.1. Introduction

Dans ce chapitre, nous présentons deux méthodes de décomposition sur deux problèmes similaires dans le même contexte d'un atelier de manufacture flexible. La premìere méthode décomposition est importante car elle constitue le point de départ et la base de nos travaux présentés dans les deux articles subséquents. Ces deux premières approches présentent chacune une méthode de décomposition hybride qui combine la programmation par contraintes (PC) et la programmation linéaire en nombres entiers (PLNE) pour résoudre des problèmes intégrés d'ordonnancement et de routage sans conflits. Nous utilisons la PC pour la partie ordonnancement car cet outil est très efficace pour résoudre des problèmes d'ordonnancement qui contiennent des contraintes logiques (ce qui est le cas dans notre problème). Le routage sans collisions est fait grâce à la PLNE. Nous avons choisi la PLNE dans cette partie suite à l'absence de modèle de PC efficace pour trouver des routes sans conflits mais aussi pour profiter de la sous structure de réseau de notre modèle. Nous faisons aussi l'ordonnancement implicite des tâches aux machines se trouvant aux points de livraison car la fin d'une livraison correspond au début de traitement du produit livré.

Dans les deux méthodes de décomposition présentées dans ce chapitre, les algorithmes que nous utilisons sont généraux car ils s'appliquent à des réseaux quelconques. Dans le premier algorithme, il n'y a pas de traitement préventif des conflits, tandis que dans le second algorithme, un début de traitement préventif des conflits est fait grâce à l'utilisation de coupes logiques pour filtrer des solutions indésirables (solutions optimales pour le modèle de PC mais qui présentent des conflits de routage). L'algorithme

présenté dans la première méthode de décomposition se comporte bien sur un nombre réduit de problèmes mais les résultats se détériorent lorsque nous introduisons des contraintes additionnelles, un grand nombre de requêtes et un horizon plus grand. En effet, un grand nombre de solutions optimales pour le modèle de PC qui présentent des conflits de routage apparaissent. Les contraintes additionnelles spécifient que : (1) d'une part, il existe des contraintes de priorité entre des tâches devant être exécutées sur des stations de travail différentes. (2) d'autre part, si deux tâches A et B sont liées par des contraintes de priorité sur une même station de travail alors aucune autre tâche ne peut être exécutée entre A et B sur la même station de travail. Supposons que A corresponde à une livraison, B corresponde à une collecte et que A précède B, le produit de A est livré dans une station de travail et la machine située dans la station de travail commence le traitement du produit A tout de suite après pour le transformer en un produit B. Alors aucune autre collecte ou livraison ne peut avoir lieu sur cette station de travail tant que le produit B n'est pas collecté. Supposons maintenant que A corresponde à une collecte, B corresponde à une livraison et que A précède B. Le produit de A doit d'abord être collecté, ce qui marque la fin de la période de traitement par la machine située dans la station de travail. Puisque la station de travail devient libre, la livraison B peut débuter.

Ainsi, il nous a paru naturel d'essayer d'éliminer une bonne partie de ces solutions indésirables grâce à une meilleure communication entre les modèles de PC et PLNE. Pour cela, nous avons introduit des coupes logiques de réalisabilité. Ce qui nous permet de résoudre beaucoup plus de problèmes avec des instances contenant jusqu'à six chariots autoguidés. Toutefois la nature très disjonctive des coupes logiques utilisées ralentit graduellement le modèle de PC à chaque itération de l'algorithme global. En outre, aussi bien le modèle de PC que celui de PLNE ont besoin d'être améliorés par une meilleure prise en compte de la structure de flux. Le modèle de PLNE présente parfois le défaut d'avoir une solution à sa relaxation linéaire sans qu'une solution entière ne soit disponible avec un temps de réponse qui peut être long à obtenir. Ce qui n'est pas intéressant car il peut y avoir un grand nombre d'itérations entre les modèles de PC et PLNE. En outre, nous avons consacré dans la deuxième méthode de décomposition une section entière à ce qui n'a pas bien marché dans nos tentatives d'améliorer l'approche qui utilise des coupes logiques de réalisabilité.

4.2. Modèle initial

Cette section porte sur des travaux qui sont à la base des algorithmes présentés dans les articles qui suivent. Ces travaux ont été présentés dans un article Corréa *et al.* (2004)) qui a été arbitré et publié dans *Lecture Notes in Computer Science (LNCS 3011)* et qui est mis en annexe.

4.2.1 Introduction

Ce travail porte sur des chariots automatiques (AGVs) dans un atelier flexible (FMS). L'atelier flexible est composé de différentes cellules appelées stations de travail, chacune ayant une fonction spécifique comme le fraisage, le lavage ou l'assemblage. Un chariot automatique est un équipement de manutention qui se déplace dans un réseau physique de chemins composés de segments reliés par des points d'intersection. Les chariots automatiques se déplacent à une vitesse constante dans le réseau physique et ne peuvent s'arrêter que sur les points d'intersection. Chaque cellule est reliée au réseau de segments par un poste d'échanges où les collectes/livraisons sont effectuées. Le réseau est bidirectionnel et les chariots automatiques peuvent se déplacer en position *marche avant* ou *marche arrière*. Les chariots doivent transporter des produits d'une cellule à une autre. La charge unitaire de chaque chariot est d'une palette et le nombre de chariots disponibles est connu. La durée de chaque collecte ou livraison est connue. Dans la mesure où plusieurs chariots circulent simultanément dans le même réseau, les collisions doivent être évitées. Il existe deux types de collisions : le premier type représente la situation où deux chariots se dirigent vers le même noeud tandis que le second type décrit la situation où deux chariots se déplacent sur un même arc dans des directions opposées. Chaque jour une liste de commandes est établie, toute commande correspondant à un produit à fabriquer (un produit signifie ici une ou plusieurs unités du même produit). Une commande détermine une succession d'opérations sur les diverses stations de travail. L'ordonnancement des tâches dans l'atelier est fait et ce plan de production fixe les temps au plus tôt de début de chaque commande, précisément le temps de début de chaque collecte et de sa livraison associée.

Ainsi des palettes de produits sont déplacées d'une station de travail à une autre. Chaque requête de transport est vue comme un couple collecte / livraison avec leurs temps au plus tôt de début. L'horizon de planification est divisé en périodes de 15 secondes. À chaque période, la position de chaque chariot doit être connue. Il y a un délai de production lorsqu'une palette est livrée après sa période au plus tôt de livraison.
Ainsi, le problème dans l'atelier flexible est défini de la façon suivante :

> *Étant donné un nombre d'AGVs disponibles et un ensemble de requêtes de transport, trouver l'affectation de requêtes aux chariots automatiques et leurs routes sans conflits qui permettent de minimiser les retards de production.*

4.2.2 Revue de Littérature

Pour une revue récente sur les problèmes sur les chariots automatiques, le lecteur est referé à la revue de Qiu *et al.* (2002). Ces auteurs ont classé les algorithmes sur les chariots automatiques en trois grands groupes :

- les algorithmes d'optimisation pour des réseaux quelconques.

- les algorithmes pour la conception des réseaux.

- les algorithmes d'optimisation pour les réseaux à topologie spécifique.

Dans cette première approche, nous travaillons sur des algorithmes du premier groupe. Les méthodes qui y sont utilisées peuvent être divisées en trois catégories : (1) les méthodes statiques où un chemin entier reste occupé jusqu'à ce que le véhicule complète

sa route ; (2) les méthodes basées sur les fenêtres de temps où un segment d'un chemin peut être utilisé par différents véhicules pendant différentes fenêtres de temps ; (3) les méthodes dynamiques où l'utilisation de n'importe quel segment de chemin est déterminé de façon dynamique pendant le routage plutôt qu'avant le routage comme dans les catégories (1) et (2). Notre méthode appartient à la troisième catégorie et nous travaillons sur des réseaux bidirectionnels, des problèmes de routage sans conflits avec une approche d'optimisation. En outre, nous avons un ensemble statique de commandes c'est-à-dire que toutes les commandes sont connues *a priori*.

Krishnamurthy *et al.* (1993) ont été les premiers à proposer une approche d'optimisation pour résoudre un problème de routage sans conflits. Leur objectif consiste à minimiser le temps de fin de la dernière tâche (*makespan*). Même si elle n'est pas optimale, leur méthode génère de très bonnes solutions (la génération de colonnes est faite au noeud racine seulement). Langevin *et al.* (1996) proposent une approche basée sur la programmation dynamique pour résoudre de façon exacte des problèmes de répartition et de routage sans conflits contenant deux chariots. Desaulniers *et al.* (2003) présentent une méthode exacte et des tests sur des instances contenant jusqu'à quatre chariots. Leur jeu de données est similaire à celui de Langevin *et al.* (1996). Toutefois leur approche présente quelques limites car l'efficacité de leur méthode dépend beaucoup de la performance de l'heuristique gloutonne de départ. Si, au début de la méthode, aucune solution réalisable n'est pas trouvée alors il n'y aura pas de solution optimale. Les performances de leur heuristique se détériorent quand le niveau de congestion augmente et quand le système considère plus de six véhicules.

4.2.3 Une approche hybride Programmation par Contraintes / Programmation Linéaire en Nombres Entiers

Les décisions sur la répartition et le routage sans conflits des chariots automatiques peuvent être décomposées en deux parties : la première partie consiste en la répartition des requêtes aux chariots et à définir les temps d'exécution des tâches de collecte / livraison. La seconde partie consiste à assurer le routage simultané des véhicules.

L'approche hybride présenté que nous présentons combine un modèle de programmation par contraintes (PC) pour trouver l'affectation des requêtes de transport aux chariots automatiques avec leurs véritables temps de collecte / livraison tout en minimisant les retards de production, et un autre modèle de programmation linéaire en nombres entiers (PLNE) pour le routage sans conflits. Les deux modèles sont intégrés dans une procédure itérative (voir figure 4.1). Pour chaque affectation et horaire trouvés par le modèle de PC, le modèle de PLNE essaie de trouver des routes sans conflits qui satisfont l'horaire. La PC est utilisée pour résoudre la première partie car elle est très efficace pour résoudre des problèmes d'ordonnancement et dans notre cas elle nous permet d'identifer toutes les solutions optimales. Des solutions optimales qui présentent le même retard total de production mais représentent des affectations différentes peuvent donner des solutions de routage très différentes. La partie routage sans conflits est formulée avec la PLNE car elle peut être modélisée avec un réseau espace-temps qui présente des sous structures qui permettent un solution rapide.

La méthode peut être décrite en trois étapes :

- Étape 1 : trouver une solution optimale $x^\star$ au modèle de PC c'est-à-dire une affectation ordonnée des requêtes aux chariots. Soit $z^\star$ la valeur optimale de la fonction objectif (retard total de production).

- Étape 2 : utiliser $x^\star$ dans le modèle de PLNE pour trouver un routage sans conflits de chaque chariot automatique. S'il en existe, la solution optimale du modèle original est trouvée. Sinon (pas de solution réalisable trouvée), aller à l'étape 3.

- Étape 3 : trouver une autre solution optimale au modèle de PC qui est différente de $x^\star$ mais avec la même valeur de la fonction objectif. S'il en existe, retourner à l'étape 2. Si aucune solution de routage réalisable n'est trouvée pour les solutions optimales du modèle de PC, aller à l'étape 1 et ajouter une borne inférieure à la fonction objectif ($f(x) > z^\star$) avant de résoudre à nouveau le modèle de PC. Cette borne inférieure est fixée à $z^\star$ et est toujours mise à jour lors du retour à l'étape 1.

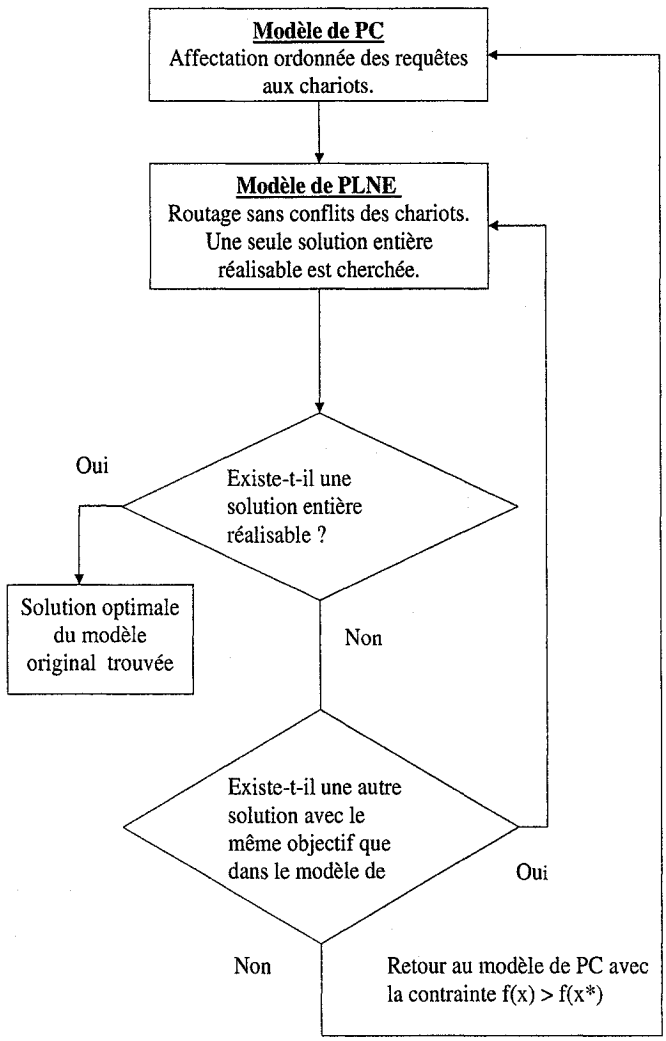

Figure 4.1 – Première décomposition

Le modèle de Programmation par Contraintes

Le modèle répond à la question de savoir quel chariot effectue quelle tâche de manutention et en quelle période en donnant une affectation ordonnée des requêtes aux chariots. Le total des retards de production est mesuré en faisant la somme des différences entre le temps de début réel et le temps de début estimé de chaque livraison. Dans ce modèle la matrice des distances (temps entre les points de collecte et les points de livraison) est obtenue en calculant les plus courts chemins entre les noeuds (points de collecte ou livraison). Ces distances calculées ne tiennent pas compte des conflits possibles, elles constituent des approximations (bornes inférieures) des délais réels entre deux tâches (collecte ou livraison) consécutives sur un même chariot automatique.

Ensembles et paramètres du modèle de PC

$DummyStartTasks$:	ensemble des tâches fictives de départ.
$Start[k]$:	noeud de départ du chariot k.
$Pickups$:	ensemble des collectes.
$SP[.,.]$:	matrice des plus courts chemins entre les points de tâches (collecte/livraison).
$Node(p)$:	point où la tâche p est exécutée.
$nbRequests$:	nombre de requêtes de transport à satisfaire.
réelles.	
$nbChar$:	nombre de chariots automatiques disponibles.
$Requests$:	ensemble des requêtes de transport.
$DummyStartRequests$:	ensemble des requêtes fictives de départ.

$Inrequest$:	ensemble des requêtes fictives de départ et des requêtes .
$Pick[.]$:	collecte associée à une requête de transport.
$Del[.]$:	livraison associée à une requête de transport.
$Duration$:	durée d'une tâche.
$Priorities$:	ensemble de couples de tâches (collecte/livraison) liées par des contraintes de priorité.
$Tasks$:	ensemble de toutes les tâches ayant un successeur.

Ce modèle utilise les trois variables suivantes :

$Alloc[i] = k$	si la tâche i est exécutée par le chariot k. Les indices inférieur à 1 représentent les requêtes fictives.
$Succ[u] = v$	si la requête v est la successeure de la requête u sur le même chariot.
$Starttime[j]$	temps de début de la tâche j.

Pour chaque chariot, un couple de tâches fictives est créé, une tâche de départ et une tâche d'arrivée. La tâche de départ a les caratéristiques suivantes : son noeud est le point de départ du chariot, sa durée et son temps au plus tôt de début sont nuls. Nous définissons par $Tasks$ l'ensemble des tâches fictives de départ ainsi que les collectes et livraisons réelles. Une requête consiste en une collecte et une livraison. La fonction objectif est la somme des différences entre le temps réel de début de chaque livraison et le temps au plus tôt (donné) de cette même livraison.

Les contraintes utilisées dans ce modèle sont les suivantes :

$$Alloc[1-k] = Alloc[nbRequests + k] \quad \forall\, k \in Char \tag{4.1}$$

$$\sum_{s \in Inrequest} (Succ[r] = s) = 1 \quad \forall\, r \in DummyStartRequests \tag{4.2}$$

$$Alloc[o] = Alloc[Succ[o]] \quad \forall\, o \in Tasks \tag{4.3}$$

$$Starttime[d] = 0 \quad \forall\, d \in DummyStartTasks \tag{4.4}$$

$$\begin{aligned}&(Alloc[d] = k) \wedge (Succ[d] = r) \Rightarrow \\ &\quad (Starttime[pick[r]] \geq SP[Start[k], node[pick[r]]) \\ &\qquad \forall\, k \in Char, d \in DummStartRequests, r \in Requests\end{aligned} \tag{4.5}$$

$$Alldifferent(Succ) \tag{4.6}$$

$$\begin{aligned}&Starttime[pick[r]] + 1 + SP[Start[k], node[pick[r]] \leq Starttime[del[r]] \\ &\quad \forall\, r \in Requests\end{aligned} \tag{4.7}$$

$$\begin{aligned}&Succ[del[r1]] = pick[r2] \Rightarrow \\ &\quad (Starttime[del[r1]] + 1 + SP[node[del[r1]], node[pick[r2]] \leq Starttime[pick[r2]]) \\ &\qquad \forall\, r1, r2 \in Requests\end{aligned} \tag{4.8}$$

$$\begin{aligned}&Starttime[before[u]] + duration[before[u]] \leq Starttime[after[u]] \\ &\quad \forall\, u \in Priorities\end{aligned} \tag{4.9}$$

$$\begin{aligned}&(Starttime[i] \geq Starttime[j] + 1) \vee \\ &\quad (Starttime[i] + 1 \leq Starttime[j]) \\ &\qquad \forall\, i, j \in Tasks : (i \neq j) \wedge (node[i] = node[j])\end{aligned} \tag{4.10}$$

Les contraintes (4.1) assurent qu'une tâche fictive de départ et la tâche fictive d'arrivée correspondante sont exécutées par le même chariot. Les contraintes (4.2) signifient que la successeure d'une requête fictive de départ est une requête réelle ou une requête fictive d'arrivée (Dans ce cas le chariot n'exécute aucune tâche durant tout

l'horizon. Toutefois, il peut bouger pour éviter les collisions). Les contraintes (4.3) assurent que chaque requête et sa successeure doivent être exécutées par le même chariot. Les contraintes (4.4) spécifient qu'au début de l'horizon (période 0), chaque chariot se trouve sur son noeud de départ. Les contraintes (4.5) assurent que chaque chariot doit avoir le temps nécessaire pour atteindre son premier point de collecte. Les contraintes (4.6) spécifient que la successeure de chaque requête est unique. Les contraintes (4.7) assurent que tout chariot qui exécute une requête doit avoir assez de temps pour aller de son point de collecte à son point de livraison. Les contraintes (4.8) signifient que si une requête est la successeure d'une autre requête sur un même véhicule, ce chariot doit avoir assez de temps pour faire le voyage à vide du point de livraison de la première requête au point de collecte de la seconde requête. Les contraintes (4.9) assurent que, pour chaque couple de tâches liées par des contraintes de priorité, la première tâche doit débuter et être complétée avant que la seconde tâche ne débute. Les contraintes (4.10) spécifient que pour chaque couple de tâches (collecte ou livraison) qui doivent être exécutées sur le même noeud, l'une doit débuter au moins une période après l'autre.

Le modèle de Programmation Linéaire en Nombres Entiers

Après importation des résultats du modèle de PC, le modèle de PLNE cherche s'il existe pour les chariots un routage réalisable sans conflits. Ce modèle de PLNE aurait pu être formulé comme un problème de satisfaction de contraintes (PSC) car nous cherchons seulement un routage réalisable sans conflits pour tous chariots. Toutefois la sous structure de réseau du problème de routage permet d'utiliser un modèle de PLNE où une seule solution réalisable est cherchée, ce qui évite la recherche potentiellement longue d'une solution optimale. Le modèle de PLNE correspond à un réseau espace - temps qui définit la position de chaque chariot à tout moment (voir figure 4.2). Le réseau physique est composé de segments de longueur 1, 2 ou 3. Ce réseau a été transformé en un réseau orienté où tous les arcs sont de longueur 1 en incorporant des noeuds fictifs sur les segments de longueur 2 ou 3. À chaque période, il y a un noeud pour chaque point d'intersection de deux segments du réseau (noeuds

fictifs étant inclus). Un arc est défini entre deux noeuds entre deux périodes successives si les noeuds d'intersection correspondants sont adjacents sur le réseau physique. Chaque chariot entre dans le réseau par un noeud spécifique à la période 0. Le modèle du réseau espace - temps présente les caractéristiques suivantes :

- une unité de flux est équivalent à un véhicule présent dans le système.

- à chaque période, le flux total entrant est égal au nombre total de chariots disponibles.

- au plus une unité de flux peut entrer dans un noeud (évitement de collisions sur les noeuds).

- il y a conservation de flux sur chaque noeud.

- un arc dont l'origine coincide avec la destination en deux périodes successives est une attente sur un noeud.

- un chariot peut se déplacer sans avoir de tâche précise à exécuter, seulement pour éviter des conflits.

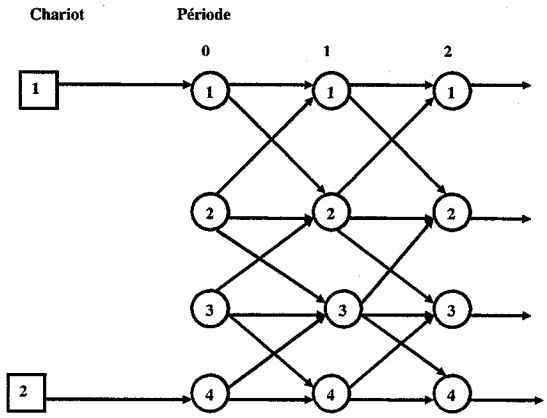

Figure 4.2 – description du réseau espace-temps

Plusieurs versions de modèle de PLNE sont présentement testées. Nous présentons ici celle qui a donné les meilleurs résultats.

Ensembles et paramètres du modèle de PLNE

$Char$:	Ensemble des chariots.
$Nodes$:	Ensemble des noeuds.
$Periods$:	Ensemble des périodes.
$ArcsPlus$:	Ensemble de tous les arcs (incluant ceux qui contiennent des noeuds fictifs).
M :	Longueur de l'horizon.

Les valeurs de $Alloc[.]$ et $Starttime[.]$ sont importées du modèle de PC. $Segment[a]$ est un enregistrement ayant deux champs : le premier champ ($Segment[a].orig$) est l'origine de l'arc a tandis que le second champ ($Segment[a].dest$) est la destination de a.

Les variables du modèle de PLNE sont :

$Y[k,t,p]=1$	si le chariot $k \in Char$ est sur le noeud $p \in Nodes$ à la période $t \in Periods$.
$Z[k,t,a]=1$	si le chariot $k \in Char$ commence la visite de l'arc $a \in ArcsPlus$ à la période $t \in [0..M-1]$.

Le modèle de PLNE est défini de la façon suivante :

$$Min \sum_{k \in Char,\, t \in Periods,\, p \in Nodes} Y[k,t,p]$$

s.c.

$$Y\left[k, Starttime\left[2-k\right], node\left[Task\left[2-k\right]\right]\right]=1 \qquad \forall\, k \in Char \tag{4.11}$$

$$Y\left[Alloc\left[r\right], Starttime\left[pick\left[r\right]\right], node\left[Task\left[r\right]\right]\right]=1 \qquad \forall\, r \in Requests \tag{4.12}$$

$$Y\left[Alloc\left[r\right], Starttime\left[del\left[r\right]\right], node\left[Task\left[r\right]\right]\right]=1 \qquad \forall\, r \in Requests \tag{4.13}$$

$$Y\left[Alloc\left[r\right], Starttime\left[pick\left[r\right]\right]+1, node\left[Task\left[r\right]\right]\right]=1 \qquad \forall\, r \in Requests \tag{4.14}$$

$$Y\left[Alloc\left[r\right], Starttime\left[del\left[r\right]\right]+1, node\left[Task\left[r\right]\right]\right]=1 \qquad \forall\, r \in Requests \tag{4.15}$$

$$\sum_{p \in Nodes} Y\left[k,t,p\right]=1 \qquad \forall\, t \in Periods, k \in Char \tag{4.16}$$

$$Y\left[k,t,Segment\left[a\right].orig\right]+Y\left[k,t+1,Segment\left[a\right].dest\right]-Z\left[k,t,a\right] \leq 1$$
$$\forall\, k \in Char, t \in \left[0,..,M-1\right], a \in Arcs \tag{4.17}$$

$$Z\left[k,t,a\right] \leq Y\left[k,t,Segment\left[a\right].orig\right]$$
$$\forall\, k \in Char, t \in \left[0,..,M-1\right], a \in Arcs \tag{4.18}$$

$$Z\left[k,t,a\right] \leq Y\left[k,t+1,Segment\left[a\right].dest\right]$$
$$\forall\, k \in Char,\ t \in \left[0,..,M-1\right], a \in Arcs \tag{4.19}$$

$$\sum_{a \in ArcsPlus} Z\left[k,t,a\right]=1$$
$$\forall\, t \in \left[0,..,M-1\right],\ k \in Char \tag{4.20}$$

$$\sum_{p \in Nodes} Y\left[k,t,p\right] \leq 1+$$
$$\left(\sum_{r \in Realtasks\,:\,t=Starttime[r] \wedge p=node[Task[r]]} 1\right) * \left(\sum_{r \in Realtasks\,:\,t=Starttime[r]-1 \wedge p=node[Task[r]]} 1\right)$$
$$\forall\, t \in Periods,\ k \in Char \tag{4.21}$$

$$\sum_{k \in Char} Z\left[k,t,a\right]+\sum_{k \in Char,\, b \in Opp[a]} Z\left[k,t,b\right] \leq 1$$
$$\forall\, k \in Char,\ t \in \left[0,..,M-1\right],\ a \in Arcs \tag{4.22}$$

Les contraintes (4.11) spécifient que chaque chariot doit être présent sur son noeud de départ au début de l'horizon (période 0). Les contraintes (4.12 - 4.13) signifient que les chariots doivent être présents sur leur point de tâche (collecte / livraison) au bon moment (au début de l'exécution de la tâche). Les contraintes (4.14 - 4.15) assurent que chaque chariot doit séjourner sur son point de tâche au moins pendant une période. Les contraintes (4.16) spécifient que chaque chariot occupe une position unique à chaque période. Les contraintes (4.17) signifient que si un chariot commence la visite d'un arc à la période t alors il va finir la visite de cette arc à la période $t+1$. Les contraintes (4.18) spécifient que si un chariot est sur un noeud à la période t, cela signifie qu'il a commencé la visite d'un arc (d'attente ou non) à la période $t-1$. Les contraintes (4.19) assurent que si un chariot est sur un noeud à la période $t+1$, cela signifie qu'il a commencé la visite d'un arc entrant à ce noeud (d'attente ou non) à la période t. Les contraintes (4.20) signifient que chaque chariot commence la visite d'un arc unique à chaque période (exception faite de la dernière). Les contraintes (4.21) interdisent la présence de deux chariots sur le même noeud, exception faite du cas où ils se croisent sur un noeud de tâche, l'un venant de finir sa tâche tandis que l'autre commence sa tâche sur la station de travail. Ce sont des contraintes d'évitement de collisions sur les noeuds. Les contraintes (4.22) sont des contraintes d'évitement de collisions sur les arcs : deux chariots ne peuvent pas se déplacer en même temps sur le même arc dans des directions opposées.

Résultats

La méthode de décomposition est implantée avec OPL Script. Nous avons comparé notre approche avec celle de Desaulniers *et al.* (2003) et nous avons gagné en flexibilité avec l'utilisation de la PC. Nous avons résolu de nouveaux problèmes contenant cinq à six chariots (les résultats affichés par Desaulniers *et al.* (2003) ne dépassaient pas l'utilisation de plus de quatre chariots). Le nombre de chariots utilisés est limité à six car le contexte étudié ne justifie pas l'utilisation d'un plus grand nombre de chariots. Toutefois des applications plus grandes comme le cas des terminaux à conteneurs utilisent des douzaines de chariots et aucun outil d'optimisation automatisé n'est disponible pour le moment. Dans sa version actuelle, la taille du modèle

Tableau 4.1 – description de trois problèmes

Problème	Nombre de Requêtes	**Nombre de relations de priorité**	**Chariot : Noeud Initial**
7	10	2	AGV1 : 15 AGV2 : 14 AGV3 : 19 AGV4 : 27
10	8	9	AGV1 : 3 AGV2 : 25 AGV3 : 23 AGV4 : 27 AGV5 : 12
15	13	7	AGV1 : 15 AGV2 : 14 AGV3 : 19 AGV4 : 27 AGV5 : 12 AGV6 : 25

de PLNE est trop grande et elle dépend beaucoup de la longueur de l'horizon. Ainsi des horizons de planification plus grands risquent d'être plus difficiles à prendre en charge. En outre, notre approche a besoin d'être testé sur des problèmes contenant un plus grand nombre de tâches ou chariots et avec des horizons fuyants.

Notre méthode est plus coûteuse en temps de calculs que celle de Desaulniers *et al.* (2003) même si les solutions trouvées sont en dessous de la limite de temps de calculs de dix minutes. Nos tests ont été faits sur un Pentium 4, 2.5 GHz. Desaulniers *et al.* (2003) ont fait leurs tests sur une station de travail SUNFIRE 4800 (900MHz). Notre approche peut être transformée en une heuristique en limitant de solution du modèle d'ordonnancement à 30 secondes (la meilleure solution rélisable trouvée en 30 secondes sera retenue).

Voici une description trois problèmes (voir tableau 4.1) et tableaux de résultats

Tableau 4.2 – problèmes à trois chariots

		Modèle de PC initial				
Problème : Nombre de chariots	**Nombre de requêtes**	**Temps de résolution**	**Choice points**	**Temps global de résolution**	**Nombre d'itérations**	**Délai de production**
1 : 3	8	2.56	17038	84.80	0	0
2 : 3	10	0.45	2585	147.72	0	0
3 : 3	8	1.39	10331	199.4850	0	0
4 : 3	9	7.11	43674	210.55	0	0
5 : 3	10	7.55	40898	140.91	0	1
6 : 3	8	0.19	1461	212.63	0	2
7 : 3	10	2.52	15638	229.35	0	2
8 : 3	8	0.25	1771	198.85	0	2
9 : 3	8	0.70	3786	234.06	0	6
10 : 3	8	0.67	5737	84.04	0	8
11 : 3	8	1.36	6689	88.35	0	10
12 : 3	7	0.11	848	45.86	0	6
13 : 3	12	298.74	1052988	534.79	20	26
14 : 3	12	101.30	530036	358.67	2	0
15 : 3	13	144.50	471880	282.16	4	25

Tableau 4.3 – problèmes à cinq chariots

		Modèle de PC initial				
Problème : Nombre de chariots	**Nombre de requêtes**	**Temps de résolution**	**Choice points**	**Temps global de résolution**	**Nombre d'itérations**	**Délai de production**
1 : 5	8	6.81	41772	140.02	0	0
2 : 5	10	0.58	2698	117.52	0	0
3 : 5	8	3.68	21899	257.95	0	0
4 : 5	9	33	203897	338.83	0	0
5 : 5	10	5.18	22146	109.10	0	1
6 : 5	8	0.44	3070	1011.13	1	2
7 : 5	10	4.64	25293	336.83	0	2
8 : 5	8	0.33	2093	194.46	0	2
9 : 5	8	-	-	-	-	-
10 : 5	8	0.06	347	89.69	0	8
11 : 5	8	0.70	4647	199.41	0	8
12 : 5	7	0.01	123	68.47	0	6
13 : 5	12	325.58	1566728	395.85	0	26
14 : 5	12	10.16	53178	625.59	16	0
15 : 5	13	519.14	2073129	706.61	4	17

Tableau 4.4 – problèmes à six chariots

		Modèle de PC initial				
Problème : Nombre de chariots	**Nombre de requêtes**	**Temps de résolution**	**Choice points**	**Temps global de résolution**	**Nombre d'itérations**	**Délai de production**
1 : 6	8	7.96	45628	84.57	0	0
2 : 6	10	0.55	2698	268.11	0	0
3 : 6	8	4.05	23544	937.97	1	0
4 : 6	9	45.80	183114	138.20	0	0
5 : 6	10	5.73	23078	78.69	0	1
6 : 6	8	0.50	3129	833.00	1	2
7 : 6	10	2.01	10903	108.55	0	2
8 : 6	8	0.36	2101	120.45	0	2
9 : 6	8	0.13	255	229.01	0	3
10 : 6	8	0.06	347	178.03	0	8
11 : 6	8	0.70	4553	93.59	0	8
12 : 6	7	0.02	123	123.50	0	6
13 : 6	12	72.31	282457	509.81	20	26
14 : 6	12	25.41	115965	455.27	20	0
15 : 6	13	525.93	2301091	605.09	0	17

Conclusion

Notre approche porte sur un algorithme hybride flexible basé sur une décomposition en une partie PC et une autre partie PLNE. Nous avons pu résoudre quelques cas d'un problème complexe de répartition et routage sans conflits de chariots non résolus auparavant. Il serait intéressant d'évaluer l'impact des contraintes de priorité entre les tâches (tant au niveau de leur nombre que de leur diversité). La suite de la thèse aborde ces questions.

4.3. Article : Scheduling and Routing of Automated Guided Vehicles : a Hybrid Approach

Dans cette section, l'article (Corréa *et al.* (2005a)) est présenté dans sa version originale. Il porte sur un problème similaire à celui traité dans la première méthode de décomposition (avec des contraintes supplémentaires) mais la méthode présentée est meilleure. Les résultats, suivis de leur discussion, y sont présentés. En outre, une section entière est consacrée à ce qui a été essayé et qui n'a pas (bien ou pas du tout) marché. Cet article a été accepté pour publication dans la revue *Computers and Operations Research.*

Scheduling and Routing of Automated Guided Vehicles : a Hybrid Approach

Ayoub Insa Corréa, André Langevin and
Louis-Martin Rousseau,

Département de mathématiques et de génie industriel, Ecole Polytechnique de Montréal, C.P. 6079, Succ. Centre-ville, Montréal (Québec) Canada H3C 3A7

Abstract

We propose a hybrid method designed to solve a problem of dispatching and conflict free routing of Automated Guided Vehicles (AGVs) in a Flexible Manufacturing System (FMS). This problem consists in the simultaneous assignment, scheduling and conflict free routing of the vehicles. Our approach consists in a decomposition method where the master problem (scheduling) is modelled with Constraint Programming and the subproblem (conflict free routing) with Mixed Integer Programming. Logic cuts are generated by the subproblems and used in the master problem to prune optimal scheduling solutions whose routing plan exhibits conflicts. The hybrid method presented herein allowed to solve instances with up to six AGVs.

Key words : Constraint programming, mathematical programming, hybrid model, logical Benders decomposition, material handling system, automated guided vehicles, vehicle routing and scheduling.

4.3.1 Introduction

This study focuses on the simultaneous scheduling and routing of automated guided vehicles (AGVs) in a flexible manufacturing system (FMS). An AGV is a material

handling equipment that travels on a network of guide paths. The FMS is composed of various cells, also called working stations, each with a specific function such as milling, washing, or assembly. Each cell is connected to the guide path network by a pick-up/delivery (P/D) station where pallets are transferred from/to the AGVs. Pallets of products are moved between the cells by the AGVs. The guide path is composed of aisle segments on which the vehicles are assumed to travel at a constant speed. The vehicles can travel forward or backward. As many vehicles travel on the guide path simultaneously, collisions must be avoided. AGV systems are implemented in various industrial contexts : container terminals, part transportation in heavy industry, flexible manufacturing systems.

For a general review on AGV problems, the reader is referred to Co and Tanchoco (1991), King and Wilson (1991) and Ganesharajah *et al.* (1998). For a recent review on AGVs scheduling and routing problems and issues, the reader is referred to the survey of Qiu *et al.* (2002). These authors identified three types of algorithms for AGVs problems : (1)for general path topology, (2) for path optimization and (3) for specific path topologies. Methods of the first type can be divided in three categories : (1a) static methods, where an entire path remains occupied until a vehicle completes its route; (1b) time-window based methods, where a path segment may be used by different vehicles during different time-windows; and (1c) dynamic methods, where the utilization of any segment of path is dynamically determined during routing rather than before as with categories (1a) and (1b). The method presented in this article belongs to the third category (1c) and addresses the conflict free routing problem with an optimization approach.

The plan of the article is as follows. Subsection 4.3.2 presents a description of the problem. Subsection 4.3.3 reviews the relevant works. Subsection 4.3.4 presents the hybrid Constraint Programming (CP)/Mixed Integer Programming (MIP) approach. Subsection 4.3.5 describes in detail the experimentation. The conclusion follows.

4.3.2 Problem description

Every day, a list of orders is given, each order corresponding to a specific product to manufacture (here, product means one or many units of the same product). The product units are carried on pallets by the AGVs and the unit load is one pallet. Each order determines a sequence of operations on the various cells (machines) of the FMS. Figure 4.3 presents the FMS with the AGV guide path used in the experimentation. The production scheduling, i.e., setting the earliest starting time of each machine operation for each pallet of each order, is done *a priori* using the P.E.R.T. method. Hence, each material handling request is composed of the pick-up and the delivery of a specific pallet with the corresponding earliest times. The guide path network is bi-directional. The vehicles can stop only at the ends (intersection nodes) of the guide path segments. There are two types of possible collisions. The first type may appear when two vehicles are moving toward the same node. The second type of collision occurs when two vehicles are traveling head-to-head on a segment. There are precedence constraints between some taks (pick-up or delivery). The duration of each pick-up or delivery is equal to one period. A production delay is incurred when a load is delivered after its planned earliest time.

The problem is thus defined as follows :

> *Given a number of AGVs (and their starting positions) and a set of transportation requests, find the assignment of the requests to the vehicles and conflict free routes for the vehicles in order to minimize the sum of the production delays.*

A solution of the problem determines :

- The number of AGVs required to perform the set of tasks.

- The assignment of requests to the AGVs.

- The position of each AGV at each period in the FMS.

- The schedule of each pick-up or delivery task.

- The revised schedule for each machine given the transportation schedule.

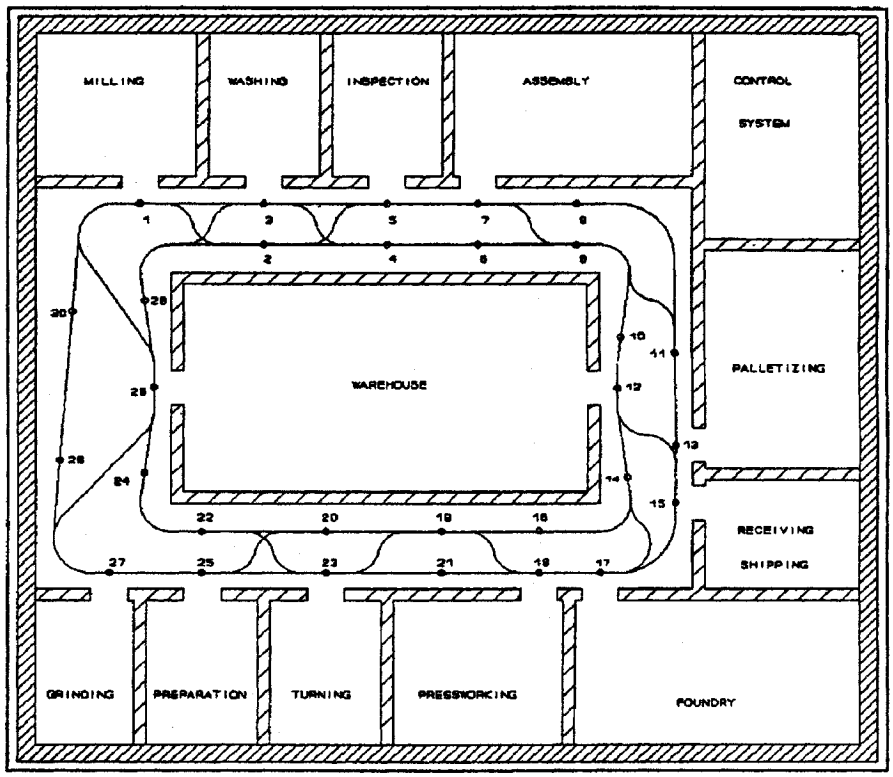

Figure 4.3 – The FMS layout with the AGV guide path

Our problem may be seen as a Vehicle Routing Problem with Time Windows (VRPTW) in which the objective is to minimize the sum of deviations from the lower bound value of the time windows of the delivery tasks subject to the following constraints :

- For each pick-up or delivery task, the lower bound of its associate time window is equal to its earliest processing time while the upper bound is the length of the horizon.

- There exist anti-collision constraints on nodes and arcs.

- There exist two types of precedence constraints between some specified pick-up and delivery tasks. The first type corresponds to a pick-up task that must immediately precede a delivery task on the same node to satisfy the production sequence. The second type corresponds to a delivery that must precede immediately a pick-up on the same node. A pallet has to be delivered and processed at a work station before being available for a pick-up.

- All nodes may be visited several times by the AGVs either to perform a task or for a simple traversal.

4.3.3 Literature review

This section is divided in three parts. In the first part, a review on AGVs scheduling and routing is presented. In the second part, a brief description of the CP paradigm is intended for the OR researchers not well acquainted with CP. The last part reviews the CP/MIP hybrid approaches.

AGVs scheduling and routing

A number of authors have addressed the conflict free routing problem with a static transportation requests set, i.e., with all requests known *a priori*. Lee *et al.* (1998)

present a two-staged traffic control scheme to solve a conflict free routing problem. Their heuristic method consists of generating off-line k-shortest paths in the first stage before the on-line traffic controller picks a conflict free shortest path whenever a dispatch command for an AGV is issued (second stage). Rajotia *et al.* (1998) propose a semi-dynamic time window constrained routing strategy. They use the notions of reserved and free time windows to manage the motion of vehicles. Krishnamurthy *et al.* (1993) propose an optimization approach. Their objective is to minimize the makespan. They assume that the assignment of tasks to AGVs is given and they solve the routing problem by column generation. Their method generates very good solutions in spite of the fact that it is not optimal (column generation is performed at the root node of the search tree only). Oboth *et al.* (1999) present a heuristic method to solve the dispatching and routing problems but not simultaneously. Scheduling is performed first and a sequential path generation heuristic (SPG) is used to generate conflict free routes. The SPG is inspired from Krishnamurthy *et al.* (1993) static version of the AGV routing problem and applied to a dynamic environment while relaxing some of the limiting assumptions like equal and constant speeds of AGVs. When conflict is encountered, no feed back is sent to the scheduling module. The AGV being routed has to be delayed if an alternate route cannot be generated. The authors use rules for positioning idle AGVs instead of letting the system manage them. Langevin *et al.* (1996) propose a dynamic programming based method to solve exactly instances with two vehicles. They solve the combined problem of dispatching and conflict free routing. Desaulniers *et al.* (2003) propose an exact method that enables to solve instances with up to four vehicles. Their approach combines a greedy search heuristic (to find a feasible solution and set bound on delays), column generation and a branch and cut procedure. Their method presents however some limits since its efficiency depends highly on the performance of the starting heuristic. If no feasible solution is found by the search heuristic, then no optimal solution can be found. The search heuristic performs poorly when the level of congestion increases.

The CP paradigm

Constraint Programming, which has been very successful in solving hard combinatorial problems in the field of scheduling, planning, and transportation (Dincbas *et al.*

(1990), Van Hentenryck (1989), Wallace (1996) has been the subject of several textbooks (Saraswat and Van Hentenryck (1995), Mariott and Stuckey (1998), Frühwirth and Abdennadher (2003), Apt (2003).

Traditionally a Constraint Programming model is composed of a set of variables (X), a set of domains (D), and a set of constraints (C) specifying which assignments of values in D to variable X are legal. The efficiency of the Constraint Programming paradigm lies in powerful constraint propagation algorithms which remove from the domain of the variables the values which will generate infeasible solutions. If constraint propagation is not sufficient to find a feasible solution then a branching process is performed to further narrow the domains ; a feasible solution is found when each domain contains only one value. The branching process is necessary in most difficult combinatorial problems. Typically, at each node of the search tree the following steps are taken : first a variable not yet fixed is selected and a remaining value of its domain is assigned to it. Then constraint propagation occurs. If during propagation the domain of a variable is emptied then the solver has detected an inconsistency in the previously taken decisions and the whole search process backtracks, typically by choosing another value for the variable. When constraint propagation terminates while there are still some unfixed variable, then the solver creates a new search node and goes on with the procedure just detailed. The branching strategy is thus defined by *variable* and *value* selection policies. This is the most simple and frequently used branching strategy but more intricate policies are often used. For instance one could split the domain of a variable into to two sets of similar size or even use two opposing constraints to define two branching directions. It is fairly simple to extend this method to solve combinatorial optimization problems, that is to identify the feasible solution which minimizes (or maximizes) a given objective function. Once a feasible solution has been identified, the set C is extended to contain a new constraint specifying that future feasible solutions should have a strictly better cost than the cost of the solution just identified. The solver thus keeps searching for better solutions until it can prove that the last found is optimal.

CP/MIP hybrid approaches

Hooker and Ottosson (2003) and Milano (2004) present comprehensive reviews on hybrid methods. We focus here on the more relevant works. Hooker (2000) gives insights on how CP can be integrated to Benders Decomposition. A number of researchers have integrated CP in the classical Benders decomposition for MIP (the master problem or the subproblem is formulated in CP and logic - *no goods* - cuts are used). Benoist *et al.* (2002) formulate their master problem as a CP model. Their master problem is reduced to a global constraint whereas the subproblems use linear duality. The global constraint uses the network structure of the original problem and consists of two types of equations defining the feasible flow problem. This method has been efficiently applied to a workforce scheduling problem in a telecommunications company call center. Eremin and Wallace (2001) also present a hybrid decomposition method in which the master problem is solved with CP. The major interest of their approach is that the user only needs to specify the variables of each subproblem. This enables the automatic derivation of the dual form of each subproblem and an automatic extraction of the Benders decomposition. It can help researchers focus on CP or mathematical programming (not both fields) for quickly designing prototypes of models.

In other papers, the master problem is solved with MIP and subproblems are formulated and solved with CP. Thorsteinnsson (2001) proposes a hybrid framework that encapsulates Benders decomposition as a special case. Jain and Grossmann (2001) use a MIP/CP decomposition method in which the master MIP model is a relaxation of the original model and feasibility subproblems are solved with CP. They proposed also a LP/CP-based branch-and-bound algorithm to solve their hybrid models. Their application example is a scheduling problem of dissimilar parallel machines. In the same line of research, the paper of Maravelias and Grossmann (2004) presents a conceptual similarity with our decomposition. Their master MIP is a relaxation of the original problem. Then given a relaxed solution, the CP subproblem checks whether there exists a feasible solution and generates integer cuts. This method is used to solve a batch chemical process problem. As an extension of the previous approach, Hooker

(2004) uses a hybrid MIP/CP decomposition method to solve a class of planning and scheduling problems for manufacturing and supply chain management. This method, named logic-based Benders decomposition, exploits the strengths of MIP in the assignment portion and CP to tackle the scheduling part. Tasks are assigned to facilities by using MIP and then scheduled subject to release dates and deadlines using CP. The cuts used are based on either the information on the sets of tasks assigned to facilities (in case of cost minimization) or the information on the makespan (in case of makespan minimization).

4.3.4 A Hybrid CP/MIP Approach

The development of our approach stems from the limitation of a previous mathematical programming approach (Desaulniers *et al.* (2003)) as discussed in Section 3.1. The basic motivation for this CP/MIP decomposition is to benefit from the strengths of CP for scheduling (as it can handle non linear constraint) and of MIP for routing in this particular context. The approach is inspired by the logic-based Benders decomposition (Hooker and Ottosson (2003)) although it does not take advantage of duality. Subsection 4.3.4.1 presents the decomposition framework, while in Subsections 4.3.4.2 and 4.3.4.3, the master and the subproblem models are respectively described. Subsection 4.3.4.4 discusses the logic cuts generated from the subproblems.

The decomposition framework

In the decomposition method developed herein, the CP master problem determines both the assignment of the transportation requests to the vehicles and the schedule (i.e., the expected times) of the pick-up and the deliveries based on the shortest path routes (neglecting the possible route conflicts). For each solution of the master problem, the MIP subproblem tries to find collision free routes satisfying the schedule obtained from the master problem. When there are no solution (i.e., it is not possible to find conflict free routes that satisfy the schedule), logic cuts are generated and sent back to the master problem. The method is depicted in Figure 4.4.

The logic cuts (described in detail in Section 4.3) are constraints added to the model to eliminate the CP solutions that have no feasible routing solution (both the incumbent and as many as possible other CP solutions with the same objective value (total delay)). One cut consists of a disjunction of three options : with the first two options, the same assignment of requests to the AGVs is kept but either the transportation time or the starting time of at least one delivery is increased by one unit, the third option consists in making a change to the assignment of requests to the AGVs (at least one request must be reassigned).

The CP master problem provides a very good lower bound to the original scheduling and routing problem and it contains essentially non linear constraints. The subproblem has a very strong minimum cost flow problem structure and thus can be solved efficiently by using CPLEX software(starting with the network simplex algorithm at the root node).

The CP model

The model determines which vehicle will be processing what material handling request at what time by generating an ordered assignment of tasks to AGVs while minimizing the total number of delivery delays. The total amount of delays is measured by summing the difference between the planned start time and the earliest start time of each delivery. In this model, the distance (time) matrix is obtained by using shortest paths between nodes. Thus, the delays calculated (which don't take into account the possible conflicts) provide a lower bound of the actual delays. A transportation request consists of a pick-up and a delivery tasks. For modeling purposes, dummy start and end requests (and tasks) are associated with each AGV. If the successor of a dummy start request is the dummy end request corresponding to the same vehicle, it means that the AGV is not used during the whole horizon. In addition, we assume that a dummy start task corresponds to a delivery task of the preceding horizon. Hence the starting (beginning of the horizon) and final (end of the horizon) positions can be seen as dummy task nodes for each AGV for the current horizon and the next one. We define the set W as the set of all tasks including the dummy start ones.

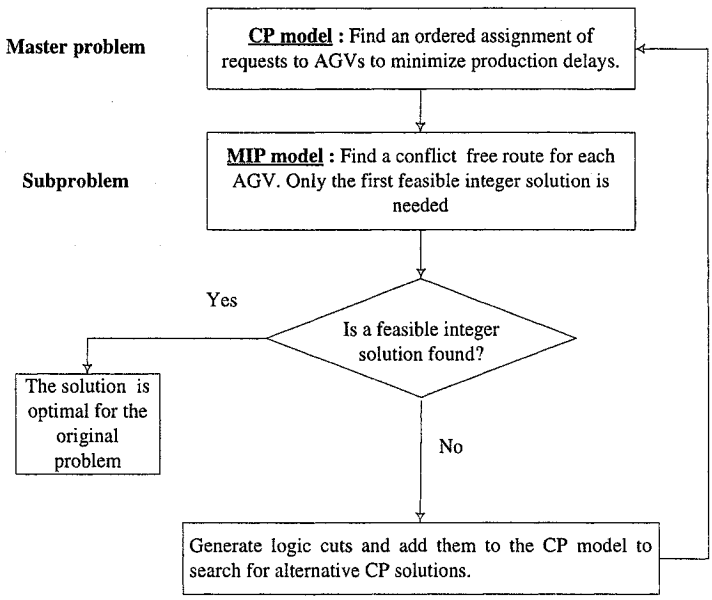

Figure 4.4 – The hybrid method

Sets and parameters of the CP model

W :	set of pick-up and delivery tasks.
W^p :	set of pick-up tasks.
W^d :	set of delivery tasks.
V :	set of AGVs.
R :	set of requests : each one is a pair of pick-up (r^p) and delivery (r^d).
T :	set of time periods.
I :	set containing R and the dummy start requests.
O :	set containing R and the dummy end requests.
P :	set of couples of tasks linked by a precedence relationship (p^1,p^2).
E :	set of all earliest start time.
$e(\cdot)$:	duration of the processing at a workstation.
$D(\cdot,\cdot)$:	length of the shortest path between the nodes of two tasks.
n_i :	node for task i.
n^v :	starting node of AGV v.
E_j :	earliest starting time of task j.

Variables of the CP model

$v_r \in V$	variable representing the AGV assigned to request r.
$s_r \in O$	variable representing the request performed immediately after r on the same vehicle.
$t_r \in T$	variable representing the start time of request r.

The objective function consists in minimizing the sum of the differences between the given earliest starting times and the actual starting time of the deliveries (the t_r variables). The constraints used in the model are the following (for concision, we have not included the starting and ending conditions which are straightforward) :

$$Min \sum_{j \in W^d} (t_j - E_j) \tag{4.23}$$

$$s.t.\quad v_o = v_{s_o} \qquad \forall\, o \in O \tag{4.24}$$

$$t_{r^p} + 1 + D(r^p, r^d) \leq t_{r^d} \qquad \forall\, r \in R \tag{4.25}$$

$$s_{r_1} = r_2 \Rightarrow t_{r_1^d} + 1 + D(r_1^d, r_2^p) \leq t_{r_2^p} \qquad \forall\, r_1, r_2 \in R \tag{4.26}$$

$$\neg(t_{p^1} \leq t_i \leq t_{p^2})$$

$$\forall\, p \in P, i \in W: (p^1, p^2 \neq i \wedge (n_i = n_{p^1} = n_{p^2})) \tag{4.27}$$

$$t_{p^1} + 1 \leq t_{p^2}$$

$$\forall\, p = (p^1, p^2) \in P : (p^1 \in W^p) \wedge (p^2 \in W^d) \tag{4.28}$$

$$t_{p^1} + 1 + e_{p^1} \leq t_{p^2}$$

$$\forall\, p = (p^1, p^2) \in P : (p^1 \in W^d) \wedge (p^2 \in W^p) \tag{4.29}$$

$$(t_i \geq t_j + 1) \vee (t_i + 1 \leq t_j)$$

$$\forall\, i, j \in W : (i \neq j) \wedge (n_i = n_j) \tag{4.30}$$

Constraints (4.24) ensure that each request and its successor are assigned to the same AGV. Constraints (4.25) specify that each vehicle processing a request must have enough time to go from the request pick-up node to the related delivery node. Constraints (4.26) ensure that if one request is the successor of another request on the same vehicle, the AGV must have enough time for dead-heading. Constraints (4.27) enforce that if two tasks linked by a priority constraint share the same node, they must be processed consecutively at this node. Constraints (4.28) state that at least one period (duration of the pick-up) must elapse between the starting times of pick-up and delivery tasks linked by a precedence constraint. Furthermore, no other task can be performed on a node between the execution of two tasks linked by a precedence constraint (more details are given in section 4.1). Constraints (4.29) ensure that, if a delivery precedes a pick-up, there is at least one period (duration of the delivery) plus the processing time of the delivered product between their starting times. Constraints (4.30) ensure that two different tasks cannot start at the same node at the same time.

Search Strategy

To help the CP model, we implemented some selection heuristics used in combination to the pre-implemented Slice-Based Search (SBS), a technique similar to Limited Discrepancy Search (LDS) (see Ilog OPL Studio 2002). In fact SBS determines the shape of the branching tree while our heuristics specify how to move inside the tree. The basic idea behind SBS (or LDS) is to minimize the number of decisions that do not respect the heuristic first choices. To implement this search, one needs a selection heuristic that ranks all branches of a node. Then at each node of the search tree, a *discrepancy* is counted each time a branch, not ranked first, is taken. The overall process thus starts by traversing the search tree without allowing any discrepancy, and then it iteratively increases the number of discrepancies tolerated and traverses the search space again. The description of our heuristic procedure is as follows : variables are chosen according to a first-fail strategy, i.e., the variable with the smallest domain is instantiated first. Regarding the order of values for the instantiation of the variables, the following strategy was used : for the v variables, similar requests (same pick-up node and same delivery node) are assigned as much as possible to the same AGV, the closest AGV to the pick-up node being chosen first. The t and s variables are instantiated to values in increasing order.

The MIP model

Since an AGV may visit an arc or a node more than once during the horizon, a time-space graph was used to model the routing of the AGVs. The time-space graph is illustrated in Figure 4.5 and can be described as follows : A node is defined for each period (of one time unit) and each endpoint of the guide path segments of the FMS. Each node of a given period is linked by arcs to the corresponding adjacent nodes of the next period. Those arcs correspond to the movement of an AGV between two adjacent endpoints of the segments of the guide path. Each node of a given period is also linked by an arc to the corresponding same node of the next period (i.e., a horizontal arc). This corresponds to waiting one unit of time at that node. There are no

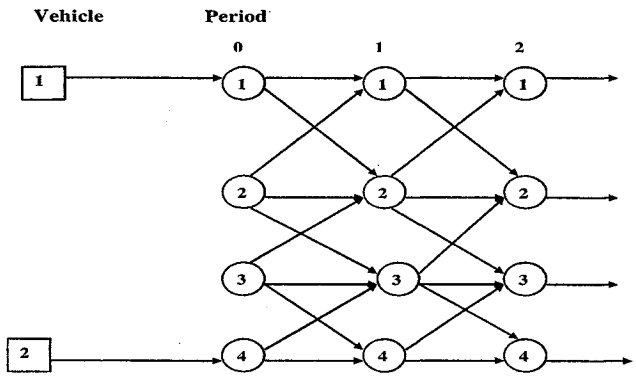

Figure 4.5 – Time-space network

arcs between nodes of a same period. All arcs of the graph have a length of one time unit. For each AGV, there is a starting node at period 0. The problem corresponds then to a multi-commodity network flow model where each commodity represents one vehicle. The master problem solution imposes that each flow visits specific nodes at specific times.

Sets and parameters of the MIP model

V :	set of AGVs.
N :	set of nodes.
T :	set of all time periods.
T^+ :	set of all time periods but the first.
A :	set of arcs.
A^+ :	set of all arcs (including waiting arcs).
$A^f[.]$:	the set of arcs coming from node i.
$A^t[.]$:	the set of arcs entering in node i.
$A^w[.]$:	the waiting arc associated to each node
$A^o[.]$:	the opposite arc of each arc.
n^v :	the starting node of AGV v.
V, T :	data obtained from the CP model. For example, T_{r^p} is the starting time of pick-up p associated to request r, T_{r^d} is the starting time of delivery d associated to request r. V_r is the AGV assigned to request r.
R :	set of requests : each one is a pair of pick-up (r^p) and delivery (r^d).
$R^+[i,t]$:	for every pair (node i, period t), set of pick-up and delivery tasks r whose service node is i and which are performed at time t $((t = t_r - 1) \wedge (i = n_r))$

Variables of the MIP model

$X^t_{k,a}$	Boolean variable =1 if AGV k starts traversing arc a at period t and 0 otherwise.

Since we only search for a feasible solution the objective function is irrelevant, we thus used a constant objective function.

The model is the following :

$$Min \quad 1 \tag{4.31}$$

$$s.t. \quad \sum_{a \in A^f[n^k]} X^o_{k,a} = 1 \quad \forall k \in V \tag{4.32}$$

$$\sum_{a \in A^+} X^t_{k,a} = 1 \quad \forall t \in T, k \in V \tag{4.33}$$

$$\sum_{a \in A^f[i]} X^t_{k,a} + \sum_{a \in A^t[i]} X^t_{k,a} = 0 \quad \forall i \in N, k \in V, t \in [1, .., (M-1)] \tag{4.34}$$

$$\sum_{k \in V} X^t_{k,a} + \sum_{k \in V} X^t_{k,A^o[a]} \leq 1 \quad \forall t \in T, a \in A \tag{4.35}$$

$$X^{T_{rp}}_{V_r, A^w[n_{rp}]} = 1 \quad \forall r \in R \tag{4.36}$$

$$X^{T_{rd}}_{V_r, A^w[n_{rd}]} = 1 \quad \forall r \in R \tag{4.37}$$

$$\sum_{k \in V, a \in A^t[i]} X^t_{k,a} \leq 1 + (\sum_{r \in R^+[i,t-1]} 1) \times (\sum_{r \in R^+[i,t]} 1) \quad \forall i \in N, t \in T^+ \tag{4.38}$$

Constraints (4.32) ensure that each AGV is located at its initial position at the beginning of the horizon while constraints (4.33) ensure that each AGV is located at a unique position at each period. Constraints (4.34) are flow conservation constraints. Constraints (4.35) are collision avoidance constraints. Constraints (4.36) state that every pick-up task must be done by the right vehicle at the right time while constraints (4.37) are the equivalent of constraints (4.36) for delivery tasks. Constraints (4.38) are node capacity constraints stating that there can be only one AGV on a node at any time except the case where one AGV leaves a node task while another one is just entering in the same node task (that's why the product of the sums is added to the right hand side of constraints (4.38)). If the above MIP has a feasible solution then optimality for the global problem is obtained. Otherwise, cuts are added to the master problem which is re-solved.

Logic cuts

As there are many different solutions to the scheduling model with the same production delay, the objective is to define cuts which forbid not only the current infeasible (i.e., conflicting) solution but as many other solutions exhibiting the same undesired properties. The logic cuts, generated when no feasible routing can be found, are based on the starting times of deliveries and the assignments of requests to AGVs. The intuition behind the presented cuts is that when a conflict occurs between two AGVs, at least one of them does not have enough time to fulfill its next task.

In addition to the parameters and sets defined in the CP model, the following ones are used :

t_j^*	the current optimal starting time of task j.
v_r^*	the current optimal vehicle assigned to r.
s_r^*	the current optimal successor of request r.

The cuts generated in this approach are essentially feasibility cuts. When a conflict arise in the routing model, it means that the duration of at least one trip (material handling or deadhead) is too short. To correct this situation it is thus necessary to either change at least one starting time of a task (pick-up/delivery) or change the assignments of transportation requests to AGVs. A logic cut consists of a disjunction of the three following constraints :

- Keep the assignment of requests to AGVs and increase the time of routing between the pick-up and the delivery of a same request. This corresponds to extending at least by one period the time window between a pick-up and a delivery. This constraints can be written as

$$\left(\sum_{r \in O}(v_r = v_r^*) = |R| + |V| \ \wedge \ \sum_{r \in R}((t_{rd} - t_{rp}) \geq (t_{rd}^* - t_{rp}^*)) \geq 1\right)$$

- OR keep the assignment of requests to AGVs and the time of routing between the pick-up and the delivery of each request but increase the starting time of some

tasks. This corresponds to delaying at least one request, namely

$$\bigvee \ (\sum_{r \in O}(v_r = v_r^*) = |R| + |V| \ \wedge \ \sum_{r \in R}(t_{rd} = t_{rd}^*) \leq |R| - 1)$$

– OR change the assignments of requests to AGVs.

$$\bigvee \ (\sum_{r \in O}(v_r = v_r^*) \leq |R| + |V| - 1)$$

4.3.5 Experimentation

This section presents the computational experiments. We first describe the instances used for the tests and then we report the results. Finally, we discuss some additional features of the method that were explored.

The instances

The FMS used for the experiments is presented in Figure 4.3. The guide path was divided into segments of 7.5 m. Assuming that the AGVs move at a constant speed of 0.5 meter per second, these segments are therefore traveled in 15 seconds, which corresponds to one time unit. The instances we used are built from the dataset of Desaulniers et al. (2003) using 12 different request sets from several orders of an order book. The details of the order book can be found in Langevin *et al.* (1996) and Drolet (1991). A planned production schedule based on average material handling times provides the processing times at the work station, the earliest processing start time, as well as the precedence relationships to satisfy for each pair of operation and part type. The number of transportation requests in those 12 sets varies from 7 to 10. There are two categories of sets, differing by the spatial density of the requests. The first category, denoted *compact*, corresponds to the case where several requests are planned in the same region of the FMS. This situation arises when an order requires the transportation of a large number of components between two or three neighbouring work stations, or when two orders require similar treatments. The second category,

called *spread out*, corresponds to the case where the requests occur in different regions of the FMS. Such a request configuration naturally leads to an assignment of the AGVs by region, thus reducing the combinatorial aspect of the problem. We have added to these 12 requests sets three other sets with more requests. This third category of request sets, denoted *mixed,* is a mixture of the types *spread out* and *compact* and might reflect more complex situations which are closer to real-world instances. Table 4.5 presents the characteristics of the fifteen problem sets tested in this paper. For each of those 15 request sets, we solved the problem with 2, 3, 4, 5, and 6 AGVs respectively, which represents 75 instances. We used a time horizon of 150 periods of 15 seconds, which corresponds to 37.5 minutes. This length of horizon was chosen in order to allow a feasible solution to all the instances. We implemented the following constraints (not present in Desaulniers et al. (2003)) in our model in order to make it more realistic :

1. When two tasks are linked by precedence constraints on the same node, no other task can be performed at this node between their starting times. For example, when a delivery precedes a pick-up on a workstation, the product delivered must be processed during a certain amount of time. And when a product is being processed in a workstation, the corresponding node can be considered busy for any other potential pick-up/delivery task.

2. When two tasks are linked by precedence constraints on two different nodes, there is an order relation between the starting times but another task can be performed between the two times at either nodes.

Table 4.6 describes in detail three request sets, one per category. The last column gives the starting position of the AGVs in the instance with 6 AGVs. For each set, the material handling requests are listed in the second column. In the third (respectively fourth) column, we have the pick-up nodes (respectively delivery nodes) and their associated earliest end of processing time (EEP) of the pallet at the pick-up node

Tableau 4.5 – Description of the problem sets

Problem set	Category	Number of requests	Number of precedence relationship
1	Compact	8	7
2	Compact	10	9
3	Compact	8	7
4	Compact	9	7
5	Compact	10	4
6	Spread out	8	1
7	Spread out	10	2
8	Spread out	8	3
9	Spread out	8	5
10	Spread out	8	9
11	Spread out	8	9
12	Spread out	7	8
13	Mixed	12	9
14	Mixed	12	6
15	Mixed	13	7

(respectively the earliest start of processing time (ESP) of the pallet at the delivery node). The fifth column presents the pallet processing time at delivery nodes. The sixth column enumerates the precedence relationship between the tasks, e.g. DX $\rightarrow$ DY means that task DX precedes task DY.

Results

Experiments were performed using OPLScript 3.6 on a Pentium 4, 1.5 GHz, 512 Mo (RAM) PC. The CP model is solved with Ilog Solver 5.2 while the MIP model is solved with CPLEX 8.0. We set arbitrarily the limit for the CPU time at 12 minutes which seems reasonable for a 37.5 minutes horizon. The results are presented in Tables 4.7 to 4.11. In those tables , 'X : Y' means that the instance considered is request set X with Y AGVs. NbCuts is the number of logical cuts generated. It corresponds to the number of iterations between the CP and MIP models. *Choicepoints* column is the number of times a variable is selected during the search. An asterisk (*) means that a solution to the original model is found after the limit of 12 minutes but within the horizon of 37.5 minutes (such solutions could be used for future fleet sizing). A hyphen (-) means that no CP solution was found in less than one hour. Looking at Tables 4.7–4.11, one

Tableau 4.6 – Detailed description of request sets 3, 15, and 13 with 6 AGVs

Set number	Request number	Pickup (Node, EEP)	Delivery (Node, ESP)	Processing time	Precedence relationship	Initial Node
3	1	(15,0)	(13,5)	12	D1 → P5	AGV1 :15
	2	(15,12)	(13,19)	12	D2 → P6	AGV2 :14
	3	(15,24)	(13,33)	12	D3 → P7	AGV3 :19
	4	(15,36)	(13,47)	12	D4 → P8	AGV4 :27
	5	(13,17)	(25,28)	10	P5 → D2	AGV5 :12
	6	(13,31)	(25,41)	10	P6 → D3	AGV6 :8
	7	(13,45)	(25,55)	10	P7 → D4	
	8	(15,24)	(13,33)	12		
15	1	(15,0)	(13,5)	10	D1 → P3	AGV1 :15
	2	(15,10)	(13,17)	10	D2 → P4	AGV2 :14
	3	(13,15)	(19,21)	24	D1 → P3	AGV3 :19
	4	(13,37)	(19,57)	24	D5 → P8	AGV4 :27
	5	(13,0)	(25,18)	40	D3 → P9	AGV5 :12
	6	(13,60)	(25,78)	40	P10 → D7	AGV6 :25
	7	(25,16)	(1,29)	10	D10 → P11	
	8	(25,79)	(1,92)	10	D11 → P12	
	9	(19,56)	(7,64)	18		
	10	(1,43)	(27,55)	10		
	11	(27,62)	(3,86)	10		
	12	(3,106)	(5,110)	20		
	13	(3,37)	(5,41)	20		
13	1	(15,0)	(13,5)	10	D1 → P5	AGV1 :15
	2	(15,12)	(13,27)	10	D2 → P6	AGV2 :14
	3	(15,24)	(13,49)	10	D3 → P7	AGV3 :19
	4	(15,36)	(13,71)	10	D4 → P8	AGV4 :27
	5	(13,25)	(19,31)	14	D5 → P9	AGV5 :12
	6	(13,47)	(19,57)	14	P6 → D3	AGV6 :25
	7	(13,69)	(19,81)	14	P7 → D4	
	8	(13,91)	(19,107)	14	D10 → P11	
	9	(19,55)	(7,100)	18		
	10	(13,0)	(25,18)	18		
	11	(25,0)	(1,18)	10		
	12	(3,30)	(5,38)	10		

can observe the following : first when the number of AGVs is not sufficient (Table 4.7, 2 AGVs), the production delays are significant. Furthermore it seems that producing a valid schedule is difficult since 6 out of the 15 problems could not be solved within one hour. By increasing the number of AGVs (Table 4.8), it becomes possible to solve all instances in the given time horizon and the added flexibility allows to reduce the production delays of most instances. If we try to increase again the number of AGVs to 4 (in Table 4.9), some instances start to experience congestion as problems 7 and 13 are now solved in about 20 minutes. The solution times reported for problem 7 show that it is easy to schedule but complex to route, which indicate that conflicts on arcs are probably difficult to avoid. Again the increase number of AGVs allows to decrease the production delays of several problems. Considering a higher number of AGVs (5 AGVs in Table 4.10 and 6 AGVs in Table 4.11) does not change this picture very much. In both situations, congestion problems prevent two instances to be solved in the given 12 minutes time horizon. Furthermore, production delays are now only slightly reduced (problem 11 in Table 4.10 and problem 9 in Table 4.11). The increase number of vehicles is thus of no help to accelerate production and even counter productive since they congest the network. For *compact* and *spread out* instances, around 90 % of time solution is consumed by the MIP model. The drawback of using a time space graph is that the number of variables and constraints of the MIP formulations rapidly grow with the size of the instance. But this representation is necessary to track the position of every AGV at every period.

Looking at the results all in all, we notice that the implemented logic cuts, are almost never generated for compact or spread out requests sets; it seems that the CP is sufficient to eliminate the conflicts on task nodes by setting good slacks for material handling of each request and deadheads between requests.

Logic cuts are mostly generated on the mixed requests sets where they allow producing potentially conflict free schedules. Unfortunately the disjunctive nature of these cuts makes the model more difficult and its resolution more time consuming. This negative impact is however limited since the number of generated cuts is rather small.

The CP search strategy we used helped to avoid a number of conflicts (particularly for *compact* or *spread out* instances) early in the scheduling stage (CP model). For

Tableau 4.7 – 2 AGVs problem set

Instance	First CP Model		Global solution time	NbCuts	Production delay
	Solution time	Choice points			
1 : 2	0.51	2057	17.45	0	0
2 : 2	0.52	1696	16.53	0	4
3 : 2	0.48	2213	102.94	0	14
4 : 2	-	-	-	-	-
5 : 2	-	-	-	-	-
6 : 2	0.28	1816	58.26	0	2
7 : 2	4.62	17187	57.25	0	2
8 : 2	-	-	-	-	-
9 : 2	0.61	3628	13.65	0	12
10 : 2	-	-	-	-	-
11 : 2	.50	2681	44.97	0	13
12 : 2	0.03	239	11.13	0	6
13 : 2	-	-	-	-	-
14 : 2	-	-	-	-	-
15 : 2	154.53	415873	212.47	1	60

compact instances, no production delay occurs with more than 3 AGVs. For *spread out* and *mixed* instances, they are all solved with 3 AGVs but with a certain level of production delay. To decrease the production delay, the number of AGVs must be increased. But this has a major drawback : it increases congestion with unsolved instances as a final result. For *mixed* problems, the scheduling problem (modeled in CP) becomes more difficult to solve. The tables of results also show that the CP model needs to be enhanced since in almost half of the instances the number of choice points (branching tree nodes) is very large, due to the lack of a more efficient search strategy for all types of problems. We mention that in each instance solved, no AGV remains idle. This is why the solution of our problem can be seen as a way of finding a balance between two objectives : increasing the number of available AGVs to decrease production delays and avoiding congestion.

Table 4.12 presents the average number of variables and constraints for the first CP and final MIP models with a specified number of AGVs available in the FMS. In these two types of models, NbVar denotes the number of variables while NbConstr is the number of constraints. The MIP model is *stable* since we have almost the same number of variables and constraints in each table shown above.

Tableau 4.8 – 3 AGVs problem set

	First CP Model				
Instance	**Solution time**	**Choice points**	**Global solution time**	**NbCuts**	**Production delay**
1 : 3	2.56	17038	84.79	0	0
2 : 3	0.45	2585	147.72	0	0
3 : 3	1.39	10331	199.48	0	0
4 : 3	7.11	43674	210.54	0	0
5 : 3	7.55	40898	140	0	1
6 : 3	0.19	1461	212.63	0	2
7 : 3	2.52	15638	229.35	0	2
8 : 3	0.25	1771	198.84	0	2
9 : 3	0.70	3786	234.06	0	6
10 : 3	0.67	5737	84.04	0	8
11 : 3	1.36	6689	88.35	0	10
12 : 3	0.11	848	45.86	0	6
13 : 3	298.74	1052988	534.79	20	26
14 : 3	101.30	530036	358.67	2	0
15 : 3	144.50	471880	282.16	4	25

Tableau 4.9 – 4 AGVs problem set

	First CP Model				
Instance	**Solution time**	**Choice points**	**Global solution time**	**NbCuts**	**Production delay**
1 : 4	5.62	33471	90.17	0	0
2 : 4	0.47	2698	195.19	0	0
3 : 4	2.58	18133	182.75	0	0
4 : 4	20.32	123212	424.41	0	0
5 : 4	4.47	20221	86.05	0	1
6 : 4	0.22	1616	203.00	0	2
7 : 4	4.62	28238	1117.05 *	1	2
8 : 4	0.44	3286	105.25	0	2
9 : 4	0.44	2301	99.84	0	5
10 : 4	0.89	7347	158.04	0	8
11 : 4	0.98	4070	129.44	0	9
12 : 4	0.01	142	75.86	0	6
13 : 4	974.21	314	1290.94 *	24	26
14 : 4	183.81	939828	602.11	24	0
15 : 4	273.10	1284629	632.27	0	17

Tableau 4.10 – 5 AGVs problem set

	First CP Model				
Instance	**Solution time**	**Choice points**	**Global solution time**	**NbCuts**	**Production delay**
1 : 5	6.81	41772	140.02	0	0
2 : 5	0.58	2698	117.52	0	0
3 : 5	3.68	21899	257.95	0	0
4 : 5	33.00	203897	338.83	0	0
5 : 5	5.18	22146	109.09	0	1
6 : 5	0.44	3070	1011.13 *	1	2
7 : 5	4.64	25293	336.83	0	2
8 : 5	0.32	2093	194.46	0	2
9 : 5	-	-	-	-	-
10 : 5	0.06	347	89.69	0	8
11 : 5	0.70	4647	199.41	0	8
12 : 5	0.01	123	68.47	0	6
13 : 5	325.583	1566728	395.85	0	26
14 : 5	10.16	53178	625.59	16	0
15 : 5	519.14	2073129	706.61	4	17

Tableau 4.11 – 6 AGVs problem set

	First CP Model				
Instance	**Solution time**	**Choice points**	**Global solution time**	**NbCuts**	**Production delay**
1 : 6	7.96	45628	84.57	0	0
2 : 6	0.54	2698	268.11	0	0
3 : 6	4.05	23544	937.97 *	1	0
4 : 6	45.79	2613777	138.20	0	0
5 : 6	5.73	23078	78.69	0	1
6 : 6	0.50	3129	833.00 *	1	2
7 : 6	2.01	10903	108.55	0	2
8 : 6	0.36	2101	120.45	0	2
9 : 6	0.13	255	229.01	0	3
10 : 6	0.06	347	178.03	0	8
11 : 6	0.70	4553	93.59	0	8
12 : 6	0.02	123	123.50	0	6
13 : 6	72.31	282457	509.81	20	26
14 : 6	25.41	115965	455.27	20	0
15 : 6	525.93	2301091	605.09	0	17

Tableau 4.12 – Some statistics on the first CP and final MIP models

NbAGV	First CP model		Final MIP model	
	NbVar	NbConstr	NbVar	NbConstr
2	75	338	42600	76400
3	83	374	63900	103100
4	89	428	85200	129800
5	96	470	106500	156400
6	101	496	127800	183100

Additional features

We report here some features that we explored, unfortunately without much success, in the hope of improving our method. We nevertheless present them as they could possibly be useful in another AGV context.

- **Search strategy based on space proximity :** This strategy assigns requests to AGVs according to the proximity of the associated pick–up node of each request to the starting node of each AGV. This strategy worked well only for problems with a *compact* set of requests (i.e., with many requests planned in the same region of the FMS).

- **Conflict detection variables :** We tried to use conflict detection variables in the MIP that would enable to send information (cuts) to the CP model. The following Boolean conflict detection variables were used :

  $C^N_{it} = 1$ if there is a conflict on node i at period t, 0 otherwise.
  $C^A_{at} = 1$ if there is a conflict on arc a at period t, 0 otherwise.

  The following non constant objective function is then minimized (the sum of conflicts on arcs and nodes) :

$$\sum_{i \in N, t \in T} C^N_{it} + \sum_{a \in A, t \in T} C^A_{at} \tag{4.39}$$

  These conflict detection variables are present in the following two families of constraints which respectively replace constraints 4.35 and 4.38 :

$$\sum_{k \in V} X^t_{k,a} + \sum_{k \in V} X^t_{k,A^o[a]} \leq 1 + C^A_{at} \quad \forall t \in T, a \in A \tag{4.40}$$

$$\sum_{k \in V, a \in A^t[i]} X^t_{k,a} \leq 1 + (\sum_{r \in R^+[i,t-1]} 1) \times (\sum_{r \in R^+[i,t]} 1) + C^N_{it} \qquad \forall i \in N, t \in T^+ \qquad (4.41)$$

However, these variables would not give enough insight about the origin of a conflict and how to repair it. When two tasks generate a conflict on a segment, it is not sufficient to simply try to remove one of the two tasks from the list of tasks assigned to a vehicle to resolve the conflict. There may exist a solution with the same assignments of tasks but with a third task delayed. In the example (Figure 4.6), the distance between node 1 and node 3 and between node 3 and node 2 is 1. The arc between node 1 and node 2 is of length 2. Here, tasks 1 and 2 cannot be delayed because their starting time is equal to their maximum starting time. But task 3 can be delayed. A conflict detection variable would detect a conflict between AGV A and AGV B even though an optimal solution may exist when delaying task 3 and allowing a detour for AGV A or AGV B.

- **Time windows for pick-ups :** In order to reduce the number of equivalent solutions to the scheduling problem, we attempted to transfer the starting time decision to the routing problem. This means that the scheduling model only defines feasible time windows for each task. These time windows are transferred to the routing problem so that they can be exploited to avoid conflicts. These time windows are intervals between the starting time and the associated maximum value of each pick-up. This strategy yielded interesting results (the presence of time windows helps the MIP model to be solved faster), but unfortunately it was no longer possible to guarantee that the precedence and consecutiveness constraints would be satisfied *a posteriori*. Precedence constraints could also be modeled in the routing model, but it was not the case of consecutiveness constraints which are not linear.

4.3.6 Conclusion

In this paper, we used a decomposition method to solve a difficult combinatorial integrated scheduling and conflict free routing problem. This hybrid method consists

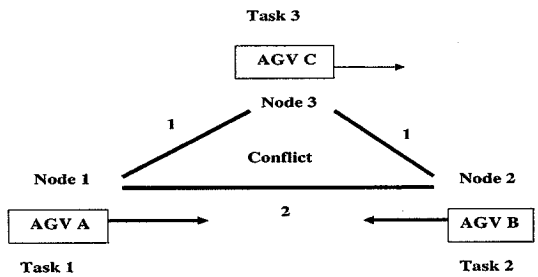

Figure 4.6 – Why conflict detection variables are not effective

in dividing the problem into two interrelated subproblems. The first subproblem is modeled with CP, a very strong tool to address scheduling problems that allowed us to design a specific search strategy. The second subproblem is basically a CSP problem modeled as a MIP problem to benefit from its network substructure. The two main reasons for choosing such a decomposition method are the need to instantiate assignment variables before routing and the existence of many non linear constraints in the scheduling part of the problem. We solved problems with up to six AGVs, thirteen requests on a rolling horizon of 37.5 minutes. Our method can also be used to determine the size of the AGVs fleet. An interesting avenue of research consists in designing better logic cuts. This would be useful, as the cuts presented herein tend to be less effective when the level of production delay increases. The design of a better search strategy for the CP model could also allow to address problems with a longer horizon, more tasks or more AGVs. In such instances, the CP model tends to perform poorly. When the domain size significantly increases, an efficient search strategy becomes essential in identifying the good values in each domain.

References

APT, K. (2003). *Principles of Constraint Programming.* Cambridge University Press.

BENOIST, T., GAUDIN, E. et ROTTEMBOURG, B. (2002). Constraint Programming Contribution to Benders Decomposition : A Case Study. *LNCS 2470. 8th International Conference, CP 2002 2470. 09/2002. Pascal Van Hentenryck, ed..* Springer.

CO, C.G. et TANCHOCO, J.M.A. (1991). A Review of Research on AGVs Vehicle Management. *Engineering Costs and Production Economics* **21** 35–42.

DESAULNIERS,G., LANGEVIN, A., RIOPEL, D. et VILLENEUVE, B. (2003). Dispatching and Conflict-Free Routing of Automated Guided Vehicles : An Exact Approach. *The International Journal of Flexible Manufacturing Systems* **15** 309–331.

DINCBAS, M., VAN HENTENRYCK, P.et SIMONIS, H. (1990). Solving Large Combinatorial Problems in Logic Programming. Journal of Logic Programming **8** 75–93.

DROLET, M. (1991). Ordonnancement d'un Carnet de Commandes pour un Atelier Flexible de Sous-Traitance. Projet de Fin d'Études, 1991. Département de Génie Industriel, Ecole Polytechnique de Montréal, Canada.

EREMIN, A. et WALLACE, M. (2001). Hybrid Benders Decomposition Algorithms in Constraint Logic Programming. *Lecture Notes in Computer Science. CP 2001. LNCS 2239.*

FRÜHWIRTH, T. et ABDENNADHER, S. (2003). Essentials of Constraint Programming. Springer Verlag.

GANESHARAJAH, T., HALL, N.G., et SRISKANDARAJAH, C. (1998). Design and Operational Issues in AGV-Served Manufacturing Systems. *Annals of Operations Research* **76** 109–154.

HOOKER, J. N. (2000). Logic-Based Methods for Optimisation. Wiley, New York.

HOOKER, J. N. (2004). A Hybrid Method for Planning and Scheduling. *ed.* Wallace, M., *CP 2004*. Springer-Verlag Berlin Heidelberg. LNCS 3258.

HOOKER, J. N. et OTTOSSON, G. (2003). Logic-Based Benders Decomposition. *Mathematical Programming* **96** 33–61.

ILOG OPL STUDIO (2002). *Ilog OPL Studio 3.6* Language Manual.

JAIN, V. et GROSSMANN, I.(2001). Algorithms for Hybrid MILP / CP Models for a Class of Optimization Problems. *Informs Journal on Computing* **13** 258–276.

KING, R.E. et WILSON, C. (1991). A Review of Automated Guided Vehicle System Design and Scheduling. *Production Planning and Control* **2** 44–51.

KRISHNAMURTHY, N.N., BATTA, R. et KARWAN, M.H. (1993). Developing Conflict-Free Routes for Automated Guided Vehicles. *Operations Research* **4** 1077–1090.

LANGEVIN, A., LAUZON, D., et RIOPEL, D. (1996). Dispatching, Routing and Scheduling of Two Automated Guided Vehicles in a Flexible Manufacturing System. *International Journal of Flexible Manufacturing Systems* **8** 246–262.

LEE, J.H., LEE, B.H. et CHOI, M.H. (1998). A Real-Time Traffic Control Scheme of Multiple AGV Systems for Collision Free Minimum Time Motion : A Routing Table Approach. *IEEE Transactions on Systems, Man and Cybernetics-Part A : Systems and Humans.* **28** 347–358.

MARAVELIAS, C. T. et GROSSMANN, I. E. (2004). Using MILP and CP for the Scheduling of Batch Chemical Processes. Régin, J-C. and Rueher, M., eds. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. First International Conference, CPAIOR. Lecture Notes in Computer Science. LNCS 3011.* Springer. Nice, France.

MILANO, M. (2004). Constraint and Integer Programming : Toward a Unified Methodology. Ramesh Sharda and Stefan Vob, eds.
OPERATIONS RESEARCH/COMPUTER SCIENCE INTERFACES SERIES.
Kluwer Academic Publishers. 33-53.

OBOTH, C., BATTA, R. et KARWAN, M. (1999). Dynamic Conflict-Free Routing of Automated Guided Vehicles. *International Journal of Production Research* **37** 2003–2030.

QIU, L., HSU, W.-J. et WANG, H. (2002). Scheduling and Routing Algorithms for AGVs : A Survey. *International Journal of Production Research* **40** 745–760.

RAJOTIA, S., SHANKER, K. et BATRA, J.L.(1998). A Semi-Dynamic Window Constrained Routing Strategy in an AGV System. *International Journal of Production Research* **36** 35–50.

SARASWAT, V. et VAN HENTENRYCK, P. (1995). Principles and Practice of Constraint Programming. 1995. The MIT Press.

THORSTEINSSON, E. S. (2001). Branch-and-Check : A Hybrid Framework Integrating Mixed Programming and Constraint Logic Programming. *Principles and Practice of Constraint Programming - CP 2001. Lecture Notes in Computer Science 2239.* 16–30.

WALLACE, M. (1996). Practical Applications of Constraint Programming. *Constraints* 1 139–168.

VAN HENTENRYCK, P. (1989). Constraint Satisfaction in Logic Programming. MIT Press, Cambridge, Mass.

CHAPITRE 5 : UN PROBLÈME INTÉGRÉ D'ORDONNANCEMENT ET DE ROUTAGE SANS CONFLITS DANS UNE MINE SOUTERRAINE

5.1. Introduction

Dans ce chapitre, nous présentons un article qui est la suite logique sur le plan algorithmique des deux premières méthodes présentés auparavant. Cet article porte sur un problème intégré d'ordonnancement et routage sans conflits dans le contexte d'une mine souterraine. Dans cet article, nous présentons une méthode de décomposition hybride qui combine la programmation par contraintes (PC) et la programmation linéaire en nombres entiers (PLNE). Nous avons conçu un algorithme basé sur la topologie spécifique des réseaux utilisés (c'est la tendance actuelle en recherche vu la difficulté des problèmes étudiés). La structure en arbre des réseaux étudiés les rend très vulnérables à la congestion. Nous profitons de cette structure spécifique pour mieux prendre en compte la congestion dans les réseaux grâce à la notion de gel de galerie. Précisons que le gel temporaire de galeries est fait pour les tâches de chargement ou déchargement seulement. Cela veut dire que dans la partie routage, rien n'empêche un chariot d'entrer dans une galerie gelée temporairement pour céder la place à un autre chariot dans une autre galerie et éviter ainsi un conflit. Dans le contexte de l'atelier flexible, l'existence de nombreuses voies de dégagement ne permet pas de développer de telles stratégies. À l'opposé des algorithmes présentés dans les deux premières méthodes de décomposition, l'algorithme conçu dans le contexte de la mine souterraine permet de faire un traitement préventif presque complet des conflits mais aussi un traitement exact de l'orientation des chariots. Parmi les caractéristiques du problème dans la mine souterraine qu'on ne trouve pas dans le problème dans l'atelier de manufacture flexible, nous pouvons citer :

- le modèle de PC fait un traitement préventif presque complet des conflits ;

- l'orientation des chariots est prise en compte car toutes les solutions obtenues doivent présenter des déplacements cohérents ;

- il n'y a pas de bornes inférieures (fournies comme données) sur les temps de début des tâches. Elles sont plutôt induites après analyse du problème ;

- il n'y a pas d'ordonnancement implicite des tâches aux machines ;

- l'algorithme conçu est adapté à la structure des réseaux (en forme d'arbre) ;

- les fonctions objectif utilisées aussi bien dans le modèle de PLNE que le modèle de PC sont différentes. Dans le contexte de la mine souterraine, la durée totale d'exécution de toutes les activités (makespan) est minimisée dans la partie ordonnancement (PC) tandis que le nombre de déplacements improductifs est limité dans la partie routage sans conflits (PLNE) ;

- l'algorithme présenté s'adapte bien aux modifications du réseau (trois réseaux étudiés). Le réseau physique est généré à chaque fois une seule fois en amont. Le code est plus aisé à maintenir (très peu de familles de contraintes à ajouter).

Dans cet article, nous utilisons des concepts d'ordonnancement par la programmation par contraintes et un meilleur modèle de PLNE. Le graphe utilisé dans le modèle de PLNE doit être vu comme le résultat de l'utilisation de deux réseaux parallèles liés au niveau de leurs noeuds d'intersection. Chaque réseau caractérise une orientation précise et le modèle de PLNE est conçu pour obtenir une forte sous-structure de réseau. Nous avons pu résoudre des problèmes avec des instances contenant jusqu'à six chariots. Suite aux résultats obtenus, nous présentons une analyse de scénarios d'agrandissement de la mine. Par la suite, nous avons pu tirer des enseignements importants sur l'origine des éventuels conflits résiduels, la congestion et par conséquent sur le dimensionnement de la flotte de chariots.

5.2. Article : Ordonnancement et routage intégrés d'une flotte de chariots dans une mine souterraine

Cet article (Corréa *et al.* (2005b)) est disponible dans les *Cahiers du Gerad* (G-2005-58, HEC Montréal, Canada.) et il a été soumis pour publication à la revue *INFOR*.

Ordonnancement et routage intégrés d'une flotte de chariots dans une mine souterraine

Ayoub Insa Corréa, André Langevin, Louis-Martin Rousseau
Department de Mathématiques et Génie industriel, École Polytechnique de Montréal and GERAD,
C.P 6079, Succ. Centre-ville Montréal, Canada, H3C 2A7,
{iacorrea, andre.langevin, louis-martin.rousseau@polymtl.ca}

Dans cet article, nous résolvons un problème intégré d'ordonnancement et de routage sans conflits d'une flotte de chariots dans une mine souterraine. Nous présentons un algorithme de décomposition hybride qui combine la programmation par contraintes (PC) et la programmation linéaire en nombre entiers (PLNE). Notre approche décompose le problème en deux parties : la première partie consiste en l'ordonnancement des tâches de chargement et déchargement de minerai tandis que la seconde partie s'occupe du routage sans conflits des chariots avec prise en compte de l'orientation des pelles mécaniques des chariots. Le réseau de galeries de la mine souterraine est en forme d'arbre. La partie ordonnancement est modélisée en PC avec un traitement préventif des conflits tandis que le routage sans conflits est modélisé en PLNE. En outre, le modèle de PLNE assure une orientation cohérente des chariots aussi bien aux points de chargement/déchargement que pendant leur routage. Nous avons testé notre algorithme sur trois réseaux de galeries différents pour analyser certains scénarios comme la croissance de la mine et la non disponibilité partielle ou totale de galeries suite à des événements imprévus (bris de chariots, chutes de roches, fuites d'eau etc.). Notre méthode de décomposition constitue un outil de dimensionnement de flotte de chariots dans une mine souterraine.

Mots-clés : méthodes hybrides ; programmation par contraintes ; ordonnancement et routage sans conflits ; orientation dans une mine souterraine ;

5.2.1 Introduction

La tendance actuelle dans l'industrie minière est une automatisation de plus en plus poussée des opérations dans les mines souterraines (or, fer, charbon, cuivre, uranium etc.). Elle est due d'une part à un manque de mineurs (problèmes de santé et sécurité au travail) et, d'autre part, à un réel besoin d'alimenter plus efficacement les hauts fourneaux et usines de traitement situés hors de la mine. Dans cet article, nous présentons une méthode de décomposition hybride pour résoudre un problème intégré d'ordonnancement et de routage sans conflits d'une flotte de chariots dans une mine souterraine. Il y a conflit quand deux chariots sont en compétition pour l'occupation d'un point ou pour parcourir un segment de galerie en même temps dans des directions convergentes. Les chariots sont soumis à des contraintes d'orientation de leurs pelles mécaniques pour accomplir leurs tâches de chargement et déchargement : chaque chariot doit toujours se présenter à son point de tâche avec une orientation appropriée de sa pelle mécanique. En outre, les réseaux utilisés sont bidirectionnels avec des galeries à voie unique. Notre approche de décomposition combine la programmation par contraintes (pour la partie ordonnancement) et la programmation linéaire en nombres entiers (pour la partie routage orienté sans conflits). La partie PC est un modèle combiné d'ordonnancement et transport qui permet un traitement préventif presque complet des conflits. La partie PLNE est un modèle espace-temps qui permet le routage sans collisions des chariots. L'orientation convenable des chariots y est assurée pour l'exécution des tâches de chargement et déchargement de minerai.

Le plan de l'article est le suivant : la section 5.2.2 présente une description du problème, suivie d'une revue de littérature à la section 5.2.3. Notre approche hybride est décrite en détail à la section 5.2.4. À la section 5.2.6, les données utilisées et les résultats obtenus sont présentés. La conclusion suit à la section 5.2.7.

5.2.2 Description du problème

La mine est présentée sous forme de trois réseaux de galeries en forme d'arbre (voir la figure 5.1). Les points de chargement et déchargement sont situés aux extrémités des

galeries (feuilles de l'arbre). Chaque point de chargement de minerai est représenté par un cercle tandis le point de déchargement est illustré par un rectangle. Dans tous les réseaux de galeries considérés, il y a un seul point de déchargement. Les réseaux sont tous bidirectionnels avec des galeries à voie unique. Les chariots peuvent être automatiques ou avec chauffeurs. L'arrêt d'un chariot n'est permis que sur les points de chargement/déchargement et d'autres points spécifiques prédéterminés. En particulier, l'arrêt n'est pas permis aux points d'intersection des galeries pour permettre la fluidité de la circulation dans la mine. La tâche de chaque chariot consiste à satisfaire des requêtes de transport. Chaque requête de transport consiste en un chargement de minerai, son transport et son déchargement. La quantité de minerai à collecter à chaque point de chargement est fixée *a priori* suite à des contraintes sur la teneur en minerai à obtenir au point de déchargement avant le traitement à l'usine. Cette quantité de minerai à collecter est exprimée en termes de nombre de voyages à faire à partir de chaque point de chargement. L'ordonnancement des tâches de chargement et déchargement est fait en tenant compte du routage des chariots. Puisque les chariots partagent le même réseau, des collisions peuvent survenir et doivent être évitées. Chaque chargement dure deux périodes tandis que chaque déchargement dure une période. Chaque chariot qui doit effectuer un chargement ou déchargement doit arriver à son point de tâche avec sa pelle orientée dans la direction appropriée c'est-à-dire vers le bout de la galerie. Notre problème est défini de la façon suivante :

> *Étant donné une flotte de chariots disponibles (avec leur position de départ associée) et le nombre de voyages à faire à partir de chaque point de chargement, il faut trouver l'affectation des requêtes de transport aux chariots et les routes sans conflits qui minimisent le temps de fin du dernier voyage (makespan) tout en assurant une orientation appropriée des pelles des chariots à chaque point de chargement ou déchargement.*

Ce problème peut être vu comme un problème de tournées de véhicules qui présente les caractéristiques suivantes :

- pour chaque noeud de tâche (chargement/déchargement), il y a un nombre obligatoire de visites à faire.

- pour chaque visite obligatoire à faire sur un noeud, le chariot utilisé doit y séjourner pendant une durée fixée.

- on peut imposer des contraintes de priorité entre les visites faites sur un même noeud.

- deux chariots différents ne peuvent visiter un même noeud à la même période (contraintes d'évitement de collisions sur les noeuds).

- deux chariots différents ne peuvent commencer en même temps la visite de certains sous-ensembles d'arcs (contraintes d'évitement de collisions sur les arcs).

- chaque chariot qui visite un noeud de tâche doit se présenter avec l'orientation appropriée.

- entre deux visites consécutives d'un noeud de tâche, certains arcs ne peuvent être visités pendant une certaine période temps (gel temporaire de galeries pour l'exécution de tâches même si un chariot inutilisé peut s'abriter dans une galerie pour céder la place à un autre chariot dans une autre galerie).

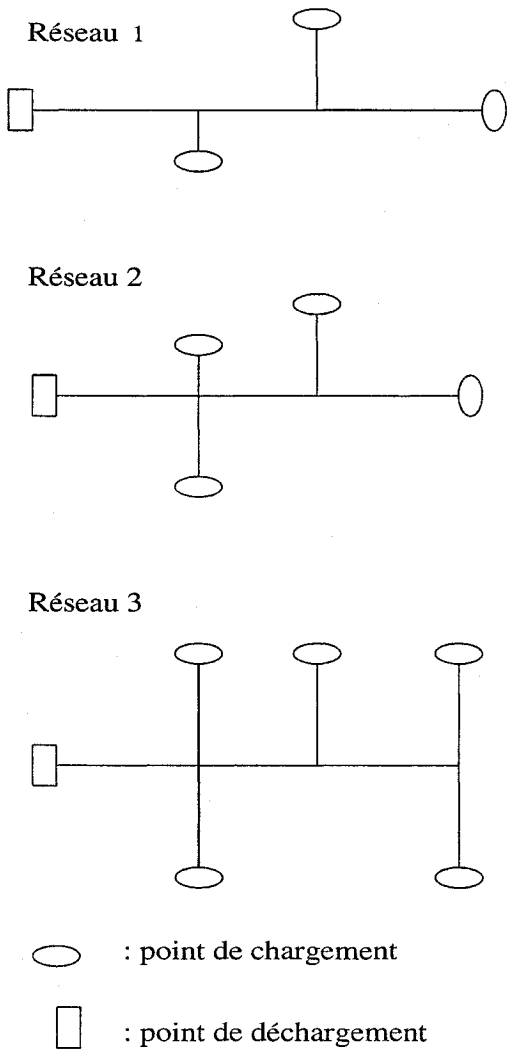

Figure 5.1 – réseaux utilisés

5.2.3 Revue de littérature

Cet article touche au problème d'ordonnancement et de routage intégrés, aux approches hybrides exactes combinant la PC et la PLNE et au traitement de l'orientation des chariots dans une mine souterraine. Aussi, cette revue porte d'une part sur les problèmes intégrés d'ordonnancement et de routage dans les contextes d'atelier flexible et mines souterraines. D'autre part, nous nous intéresserons au thème spécifique (et récent) des méthodes hybrides qui combinent la programmation par contraintes et la programmation linéaire en nombres entiers. Le lecteur est référé à Co et Tanchoco (1991), King et Wilson (1991), Ganesharajah *et al.* (1998) et Qiu *et al.* (2002) pour une revue générale sur les chariots automatiques. D'après la classification de Qiu *et al.* (2002), les algorithmes sur les chariots automatiques peuvent être classés en trois grands groupes :

- les algorithmes d'optimisation pour des réseaux quelconques.

- les algorithmes pour la conception des réseaux.

- les algorithmes d'optimisation pour les réseaux à topologie spécifique.

Peu de travaux ont été faits dans le domaine des problèmes d'ordonnancement et de routage intégrés résolus par des méthodes exactes. Langevin *et al.* (1996) ont proposé une méthode basée sur la programmation dynamique pour résoudre de façon exacte des instances à deux chariots dans le contexte d'un atelier flexible. Ensuite, Desaulniers *et al.* (2003) ont proposé une méthode exacte et ont résolu des problèmes utilisés par Langevin *et al.* (1996). Leur approche combine un algorithme glouton (pour trouver une solution réalisable et fixer des bornes sur les délais), la méthode de génération de colonnes et une procédure de Branch-and-Cut. Toutefois, leur méthode présente des limites car son efficacité dépend beaucoup de la performance de l'heuristique glou-

tonne de départ ; en effet, si aucune solution de départ n'est trouvée, alors il n'y aura pas de solution optimale. En outre, l'heuristique gloutonne perd beaucoup de son efficacité au fur et à mesure que le niveau de congestion augmente. Ils ont affiché des résultats pour des problèmes contenant jusqu'à quatre chariots. Par la suite, Corréa *et al.* (2005) ont résolu de façon exacte un problème apparenté à celui de Desaulniers *et al.* (2003). Ils ont utilisé une méthode de décomposition hybride combinant la PC et la PLNE pour résoudre des problèmes contenant jusqu'à six chariots automatiques. Les limites principales de leur approche se trouvent au niveau de leur modèle de PC et de la nature des coupes utilisées. En effet, leur modèle de PC a besoin d'être amélioré pour prévenir les conflits en amont et pour plus d'efficacité quand les requêtes de transport sont très diversifiées. Les coupes logiques ralentissent le modèle de PC du fait de leur nature disjonctive (beaucoup de noeuds de branchement de l'arbre de recherche sont créés). D'après la classification de Qiu *et al.* (2002), les travaux de Desaulniers *et al.* (2003) et Corréa *et al.* (2005) peuvent être classés dans la première catégorie d'algorithmes. Vu la complexité des problèmes intégrés d'ordonnancement et routage sans conflits, la tendance actuelle est à la conception d'algorithme de la troisième catégorie. Notre approche est une méthode de décomposition hybride qui fait partie de ce troisième groupe.

Dans le contexte des mines souterraines, peu de travaux ont été faits sur les problèmes intégrés d'ordonnancement et de routage sans conflits avec prise en compte de l'orientation des chariots. Parmi les rares auteurs qui se sont attaqués à l'orientation des chariots, il faut citer Gamache *et al.* (2004) et Bigras et Gamache (2005). Toutefois l'approche de Gamache *et al.* (2004) n'est pas exacte et son graphe ne tient pas totalement en compte l'orientation des chariots sur les arcs d'intersection. Les travaux de Bigras et Gamache (2005) portent essentiellement sur le traitement exact de l'orientation pendant le calcul des plus courts chemins et pour le routage des chariots. L'orientation des chariots est importante car dans le contexte d'une mine souterraine, le réseau est si exigu qu'un chariot ne peut pas se repositionner une fois arrivé à son point de tâche. Comme dans Bigras et Gamache (2005), notre modélisation de la partie routage orienté sans conflits s'inspire des travaux de Krishnamurthy *et al.* (1993) et Vagenas (1991). Ces auteurs ont introduit des modèles qui éclatent les points d'intersection en trois ou quatre noeuds, ce qui permet de bien identifier

l'orientation des chariots. Notons aussi que Beaulieu et Gamache (2004) ont proposé une stratégie globale basée sur la programmation dynamique et qui prend en compte l'orientation des chariots pour résoudre en temps réel des problèmes contenant jusqu'à quatre chariots. Deux réseaux utilisés sur trois contiennent des cycles. Toutefois, leur approche donne des résultats plus satisfaisants que lorsque la stratégie chariot par chariot, implantée par Bigras et Gamache (2002), est utilisée en amont. Même avec cette approche en deux étapes, des efforts supplémentaires doivent être faits pour améliorer la qualité de la solution obtenue. Les travaux de Bigras et Gamache (2002), Beaulieu et Gamache (2004) et Gamache *et al.* (2004) sont des approches en temps réel. Toutefois, le traitement de l'orientation qu'ils ont fait dépasse le cadre opérationnel : ils ont montré qu'il est possible de construire des réseaux qui assurent une orientation appropriée des chariots. Dans leurs approches, les tâches sont considérées selon leur ordre d'arrivée et il faut construire des routes sans conflits qui donnent une orientation correcte des chariots aux points de tâches.

L'approche proposée dans cet article est une méthode de décomposition hybride basée sur la coopération de solveurs de la PC et la PLNE. Le lecteur est référé au livre de Hooker (2000), à l'article de Hooker et Ottosson (2003) et au livre de Milano (2004) pour une revue détaillée des différents types de méthodes hybrides et des perspectives dans ce domaine. Les méthodes hybrides de la PC et la PLNE constituent un domaine de recherche récent où diverses directions de recherche sont actuellement explorées. Ces approches visent surtout à renforcer les outils classiques de la recherche opérationnelle mais aussi à tester les cadres théoriques proposés.

5.2.4 Une approche hybride PC / PLNE

Nous présentons ici une méthode de décomposition essentiellement basée sur la coopération de solveurs de la PC (Ilog Solver et Scheduler) et de la PLNE (CPLEX). Dans cette décomposition, la PC détermine l'affectation des requêtes de transport aux chariots et les temps de début des chargements et déchargements. Aussi, le modèle de PC évite certains conflits dans les galeries de chargement ou déchargement. Une galerie de chargement (respectivement de déchargement) est une galerie dont l'une des

extrémités est un point de chargement (respectivement de déchargement). Une galerie intermédiaire est une galerie qui n'est ni de chargement ni de déchargement. Pour chaque solution trouvée dans la partie PC, la PLNE cherche un routage sans conflits des chariots avec orientation appropriée. Quand il n'y a pas de routage, une solution alternative du modèle de PC est testée. Quand il n'y a plus de solutions alternatives, la borne inférieure de la durée totale des tâches (makespan) est augmentée d'une unité et le processus reprend. Il faut préciser que le réseau mathématique est généré une seule fois avant la première utilisation du modèle de PC et les plus courts chemins entre les noeuds sont précalculés et utilisés comme des données d'entrée par le modèle de PC. Les plus courts chemins prennent en compte le changement d'orientation du chariot entre l'origine et la destination. La méthode est illustrée à la figure 5.2. Le modèle de PC est décrit à la section 5.2.4.1 tandis que le modèle de routage est décrit à la section 5.2.4.2. Les parties PC et PLNE sont riches sur le plan de la modélisation ; avec l'utilisation de Ilog Scheduler, la partie ordonnancement s'occupe de prévenir une grande quantité de conflits qui peuvent survenir dans la partie routage. Le traitement de l'orientation des chariots est assuré grâce à un étiquetage spécifique des noeuds du graphe mathématique associé au réseau physique.

5.2.5 Le modèle de PC

Le modèle de PC est un modèle intégré de transport et ordonnancement résolu avec Ilog Scheduler 6.0. Il offre l'avantage de faire un traitement préventif des conflits avec la notion de gel temporaire de galeries. Il donne une affectation ordonnée des requêtes de transport aux chariots et les temps de début des chargements et déchargements. L'utilisation de Ilog Scheduler permet une modélisation du problème avec les caractéristiques suivantes :

- chaque requête de transport est vue comme une activité composée de trois sous-activités : le chargement de minerai, son transport et son déchargement.

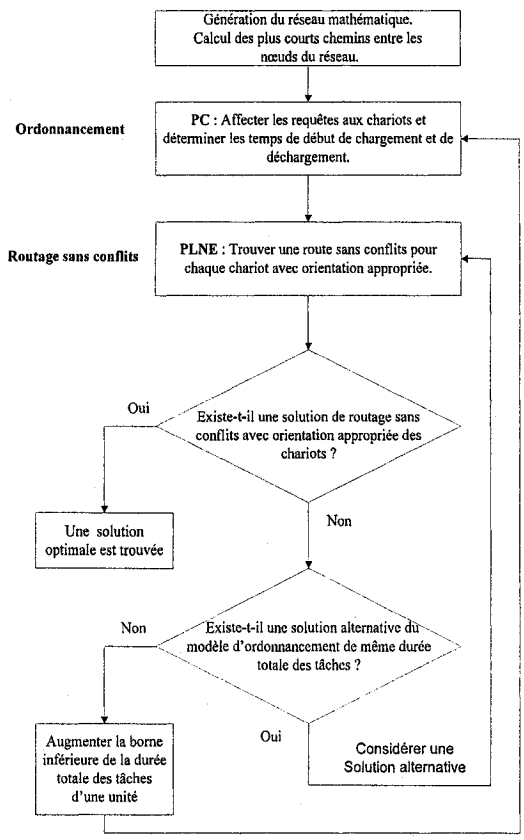

Figure 5.2 – algorithme hybride

- les temps de transition minimaux entre deux chargements consécutifs au même point mais aussi entre deux déchargements consécutifs (voyages à vide) sont modélisés efficacement. Par exemple, le temps de transition minimal entre deux chargements (respectivement déchargements) consécutifs est le temps nécessaire pour permettre au premier chariot de parcourir toute la galerie de chargement, faire le chargement (respectivement de déchargement), parcourir à nouveau la galerie de chargement (respectivement de déchargement) en sens inverse et puis permettre au second chariot de parcourir la galerie chargement (respectivement de déchargement).

- les contraintes de priorité entre certaines activités sont modélisées facilement.

- une procédure de recherche qui utilise des algorithmes très efficaces d'ordonnancement par la PC et des connaissances empiriques basées sur l'analyse de l'application est disponible.

- la galerie de déchargement est modélisée comme une *ressource unaire renouvelable.* Les chariots sont des *ressources unaires renouvelables alternatives.* Une ressource unaire est une ressource qui ne peut être utilisée en même temps par deux tâches. Elle est dite renouvelable si à la fin d'une utilisation, elle devient disponible pour une autre utilisation. Des ressources sont dites alternatives si chacune d'elles peut exécuter la même tâche.

- grâce à l'utilisation des temps de transition minimaux, toutes les galeries (chargement/déchargement) sont modélisées comme des ressources unaires renouvelables. Ainsi, des contraintes globales de la PC comme *alldifferent*, *requires*, *precedes* et *activityhasSelectedResource* sont utilisées. Une contrainte globale est une contrainte qui encapsule un ensemble de contraintes simples.
  Cet ensemble de contraintes simples est traité de façon simultanée par un algo-

rithme spécialisé de réduction de domaines.

Le lecteur est referé à Baptiste *et al.* (2001) et Ilog OPL Studio (2004) pour plus de détails sur l'utilisation de Ilog Scheduler.

Le gel temporaire de galeries est fait pour les tâches de chargement ou déchargement seulement. Cela veut dire que dans la partie routage, rien n'empêche un chariot d'entrer dans une galerie gelée temporairement pour céder la place à un autre chariot dans une autre galerie et éviter ainsi un conflit.
Dans le modèle de PC, la première requête de chaque chariot est une requête fictive associée au point de départ du chariot. Les points de chargement/déchargement sont tous numérotés. En particulier, le point de déchargement est toujours numéroté 1. Chaque activité est associée à trois variables : son temps de début, sa durée (si elle n'est pas fixée comme dans notre problème) et son temps de fin. Nous avons déduit de l'analyse du problème une borne supérieure et une borne inférieure du temps de fin de toutes les tâches de chargement et déchargement. La borne supérieure (M) est la somme des longueurs des plus courts chemins entre tous les points de chargement et le point de déchargement (voyage à vide de retour pris en compte) si toutes ces tâches étaient exécutées de façon séquentielle par un seul chariot. La borne inférieure (B) est la somme de deux termes. Le premier terme correspond au temps minimal entre deux déchargements. Le deuxième terme correspond au temps que met le chariot le plus proche de son premier point de déchargement pour parcourir le plus court chemin entre le premier point de chargement et le point de déchargement. L'orientation des chariots a été prise en compte dans l'étiquetage des noeuds et le calcul (fait en amont) des plus courts chemins entre les noeuds.

Ensembles et paramètres du modèle de PC

P :	ensemble des noeuds de chargement.
V :	ensemble des chariots.
n^v :	noeud de départ du chariot v.
$D(\cdot,\cdot)$:	matrice des plus courts chemins entre tout couple de noeuds.
R :	ensemble des requêtes réelles de transport.
I :	ensemble des requêtes réelles de transport et des requêtes fictives de départ.
h :	durée de chaque chargement.
d :	durée de chaque déchargement.
$delai$:	délai minimal entre 2 chargements consécutifs.
l_1 :	longueur de la galerie de déchargement.
S :	ensemble de toutes les tâches de déchargement.
N_c^p :	nombre de chargements à faire au noeud de chargement p.

M :	borne supérieure du début de chaque chargement avec $$M = 2 * \sum_{p \in P} (D(p,1) * N_c^p)$$
$t^r(\cdot,\cdot)$:	matrice des temps de transitions minimaux entre 2 requêtes consécutives sur le même chariot.
$t^l(\cdot,\cdot)$:	matrice des temps de transitions minimaux entre 2 déchargements consécutifs.
B :	borne inférieure du temps de fin du dernier déchargement avec $B = (\lvert S\rvert - 1) * (2 * l_1) +$ $\min_{k \in V, p \in P}(D(n^k, p) + h + D(p,1) + d)$
$tr\,[r]$:	numéro associé à la requête r.
$UnaryResource\ c[V](t^r)$	chariots munis de leur matrice t^r de transitions (voyages à vide).
$AlternativeResources\ vehicle\ (c)$	chariots définis comme des ressources unaires alternatives.
$UnaryResource\ g(t^l)$	la galerie de déchargement est une ressource unaire munie de la matrice de transitions t^l.

Variables du modèle de PC :

Activity $Makespan(0)$	le temps de fin du dernier déchargement.
Activity $Request[r \in R]$ $(h + d + D(n_r, 1))$	requête de transport (chargement, transport, déchargement).
Activity $Pick[r \in R]$ (h)	sous-activité de chargement de durée h associée à la requête r. À chaque chargement Pick[r] est associé une variable début de chargement Pick[r].start et une variable fin de chargement Pick[r].end.
Activity $Del[r \in R]$ (d)	sous-activité de déchargement de durée d associée à la requête r. À chaque déchargement Del[r] est associé une variable début de déchargement Del[r].start et une variable fin de déchargement Del[r].end.
$v_r \in V$	variable représentant le chariot affecté à la requête r.
$s_r \in O$	variable représentant la requête satisfaite immédiatement après la requête r sur le même chariot.

La fonction objectif du modèle de PC consiste en la minimisation du temps de fin du dernier déchargement.

Le modèle de PC s'écrit comme suit :

$$Min \quad Makespan.end$$

$$s.c.$$

$$B \leq Makespan.end \leq M \tag{5.1}$$

$$v_r = v_{s_r} \qquad \forall\, r \in I \tag{5.2}$$

$$alldifferent(s) \tag{5.3}$$

$$s_{r_1} = r_2 \Rightarrow Del[r1].end + D(1, n_{r2}) \leq Pick[r2].start \qquad \forall\, r_1,\ r_2 \in R \tag{5.4}$$

$$Request[r] \text{ requires vehicle} \qquad \forall\, r \in R \tag{5.5}$$

$$Request[r].start = Pick[r].start \qquad \forall\, r \in R \tag{5.6}$$

$$Request[r].end = Del[r].end \qquad \forall\, r \in R \tag{5.7}$$

$$s_r \neq r \qquad \forall\, r \in R \tag{5.8}$$

$$Pick[r].start \geq \max(D(n^{v_r}, n_r), (tr[r] - 1) * (delai + h)) \qquad \forall\, r \in R \tag{5.9}$$

$$Del[r].start \geq (tr[r] - 1) * (delai + h) + D(1, n_r) \qquad \forall\, r \in R \tag{5.10}$$

$$Pick[r] \text{ precedes } Del[r] \qquad \forall\, r \in R \tag{5.11}$$

$$Pick[r].end + D(1, n_r) \leq Del[r].start \qquad \forall\, r \in R \tag{5.12}$$

$$Del[r] \text{ requires } g \qquad \forall\, r \in R \tag{5.13}$$

$$Request[r] \text{ precedes } Makespan \qquad \forall\, r \in R \tag{5.14}$$

$$Pick[r] \text{ precedes } Pick[r+1] \qquad \forall\, r \in [1, .., (|S| - 1)] : n_r = n_{r+1} \tag{5.15}$$

$$Pick[r].end + delai \leq Pick[r+1].start \qquad \forall\, r \in [1, .., (|S| - 1)] : n_r = n_{r+1} \tag{5.16}$$

$$Del[r] \text{ precedes } Del[r+1] \qquad \forall\, r \in [1, .., (|S| - 1)] : n_r = n_{r+1} \tag{5.17}$$

$$Del[r].end + 2 * l_1 + 1 \leq Del[r+1].start$$
$$\forall\, r \in [1, .., (|S| - 1)] : n_r = n_{r+1} \tag{5.18}$$
$$Request[r] \text{ precedes } Request[r+1]$$
$$\forall\, r \in [1, .., (|S| - 1)] : n_r = n_{r+1} \tag{5.19}$$
$$activityHasSelectedResource(Request[r], vehicle, c[k]) \leftrightarrow v_r = k$$
$$\forall\, k \in V, r \in R \tag{5.20}$$

La contrainte (5.1) assure que la solution trouvée sera dans l'horizon défini. Les contraintes (5.2) signifient que toute requête de transport et sa successeure doivent être satisfaites par le même chariot. Les contraintes (5.3) imposent que chaque requête de transport est satisfaite par un unique chariot, c'est-à-dire tous les successeurs sont différents. Les contraintes (5.4) déclarent que si deux requêtes se succèdent sur un même chariot alors ce chariot doit avoir assez de temps pour faire le voyage à vide. Les contraintes (5.5) assurent que chaque requête requiert un chariot (ressource unaire renouvelable). Les contraintes (5.6) signifient que le temps de début d'une requête est égal au temps de début de son chargement associé. Les contraintes (5.7) signifient que le temps de fin d'une requête est égal au temps de fin de son déchargement associé. Les contraintes (5.8) précisent qu'aucune requête ne peut être successeure d'elle-même. Les contraintes (5.9) signifient que, pour qu'un chargement puisse être fait, il faut que le chariot utilisé ait assez de temps pour arriver à son point de chargement et que les chargements antérieurs soient d'abord faits. Ceci nous permet d'éviter d'introduire des ressources unaires supplémentaires pour les galeries de chargement. Les contraintes (5.10) signifient que, pour qu'un déchargement puisse être fait à partir d'un point de collecte, il faut que les précédents déchargements soient au préalable effectués et qu'il y ait assez de temps pour le transport. Les contraintes (5.11) précisent que tout chargement doit précéder son déchargement associé. Les contraintes (5.12) assurent que, entre la fin d'un chargement et le début de son déchargement associé, il y a assez de temps pour le transport. Les contraintes (5.13) signifient que tout déchargement requiert l'utilisation de la galerie de déchargement (ressource unaire). Les contraintes (5.14) imposent que toute requête soit satisfaite avant la fin

du dernier déchargement. Les contraintes (5.15) sont des contraintes de priorité entre les chargements. Les contraintes (5.16) imposent une période de transition minimale entre deux chargements consécutifs au même endroit. Les contraintes (5.17) sont des contraintes de priorité entre les déchargements. Les contraintes (5.18) imposent une période de transition minimale entre deux déchargements. Les contraintes (5.19) sont des contraintes de priorité entre deux requêtes consécutives. Elles permettent d'ordonner les tâches lorsqu'elles sont combinées aux contraintes (5.11), (5.15) et (5.17). Les contraintes (5.20) sont des contraintes de liaison entre la partie transport (variables v) et la partie ordonnancement du modèle de PC. Pour des raisons de concision de la présentation, nous n'avons pas inclus les contraintes touchant les requêtes fictives de départ et arrivée.

La notion de gel de galeries est prise en compte de façon implicite dans les déclarations des ressources unaires c, *vehicle* et g. Ces ressources unaires sont munies de paramètres t^r et t^l qui spécifient la période de transition minimale entre deux utilisations consécutives de la même ressource. L'activité *Del* utilise la ressource unaire g tandis que l'activité *Request* utilise l'ensemble de ressources unaires alternatives *vehicle*. En fait, avec l'activité *Request*, on a modélisé de façon implicite la contrainte logique qui impose que c'est le même chariot qui fait le chargement, le transport et le déchargement d'une même requête de transport.

Stratégie de recherche (PC)

Une stratégie de recherche en profondeur combinée à une heuristique pour spécifier la façon de parcourir l'arbre de recherche est utilisée. Dans l'heuristique, le choix des variables et activités est fait dans l'ordre suivant : v (affectation des requêtes de transport aux chariots), puis *Pick* (chargement de minerai), ensuite *Del* (déchargement de minerai). Les valeurs de v sont choisies en fonction de la proximité aux points de chargement. Les chargements et déchargements doivent commencer le plus tôt possible (on prend toujours la valeur minimale du domaine courant de chaque variable associée à ces activités). En outre, sur chaque point de chargement, on essaie

d'abord d'affecter deux chargements consécutifs à des chariots différents (les voyages sont longs et on veut rapprocher le plus possible les chargements pour obtenir un haut débit dans la mine et minimiser le makespan). Toutes les activités sont ordonnées sur les ressources unaires (galerie de déchargement, chariots) qu'elles requièrent. Les critères utilisés pour ordonner les sous-ensembles d'activités sont leurs temps au plus tôt de début, temps au plus tard de fin et la durée de toutes les activités. Les ressources les plus contraintes sont choisies en premier.

Le modèle de PLNE

À partir d'une solution du programme de PC précédent qui fournit l'affectation des requêtes aux chariots et le temps de début des chargements et déchargement, le modèle de PLNE présenté dans cette section trouve, si elle existe, une solution de routage sans conflits ainsi que l'orientation exacte des chariots. À partir d'un réseau physique, un réseau mathématique est construit, puis utilisé dans un graphe espace-temps. Un graphe espace-temps est nécessaire pour pouvoir localiser la position de chaque chariot à tout moment.

Dans ce qui suit, nous expliquons la construction du réseau mathématique pour le réseau physique numéro 2 (voir figure 5.1). Dans ce réseau physique, les galeries sont subdivisées en segments d'égale longueur et chaque segment est modélisé par deux arêtes. Chaque arête a deux noeuds qui définissent la même orientation des chariots (notée + si le chariot est orienté dans une direction, notée − dans l'autre direction). Ainsi, deux réseaux parallèles sont obtenus, l'un exclusivement avec des noeuds +, l'autre uniquement avec des noeuds − (voir figure 5.4). Les deux réseaux sont liés uniquement au niveau des points d'intersection. Dans chaque réseau, un mode de déplacement unique est considéré (chariot orienté + ou chariot orienté −). Pour chacun de ces deux réseaux (+ et −), si quatre (resp. trois) galeries se rencontrent en un point alors ce point d'intersection est éclaté en quatre (resp. trois) noeuds adjacents (noeuds dessinés en pointillés). L'arrêt sur les noeuds des intersections n'est pas permis et des points d'attente spécifiques sont choisis. Entre deux tâches consécutives (chargement/déchargement ou déchargement/chargement), tout chariot doit

changer d'orientation en passant au travers d'une intersection. La figure 5.3 illustre comment un chariot change d'orientation. Ainsi un chariot qui quitte le point A doit passer par le point B pour se retrouver en C avec l'orientation contraire de celle qu'il avait au point A. Ceci correspond, sur le réseau mathématique associé, au chemin $A, I_1, I_2, B, I_2, I_3, C$.

Le graphe espace-temps est construit en créant pour chaque période de temps un ensemble de noeuds correspondant à tous les noeuds du réseau mathématique (voir figure 5.5). Dans le graphe, il existe les types de noeuds suivants :

- les noeuds terminaux (feuilles de l'arbre). Exemple : 1^+ et 1^-.

- les noeuds d'intersection. Exemple : 4^+, 4^-, 13^+ et 13^-.

- les noeuds internes. Ce sont les noeuds qui ne sont ni des noeuds d'intersection ni des noeuds terminaux. Exemple : 2^+ et 2^-.

Chaque arête du réseau mathématique est transformée en deux arcs dans le graphe espace-temps pour chaque couple de périodes consécutives. Les arcs d'une période à l'autre correspondent au déplacement sur un segment du réseau physique ou à une attente en un noeud spécifique (voir figures 5.6 et 5.7; les arcs horizontaux correspondent à l'attente en un noeud). D'autre part, pour modéliser le passage à une intersection, des arcs de longueur nulle entre deux noeuds d'une intersection sont créés. Les figures 5.8 et 5.9 illustrent quelques arcs d'intersections du réseau 2. Des contraintes dans le modèle mathématique empêchent l'utilisation par un chariot de plus d'un arc d'intersection à chaque période, ce qui élimine la possibilité d'attente à une intersection.

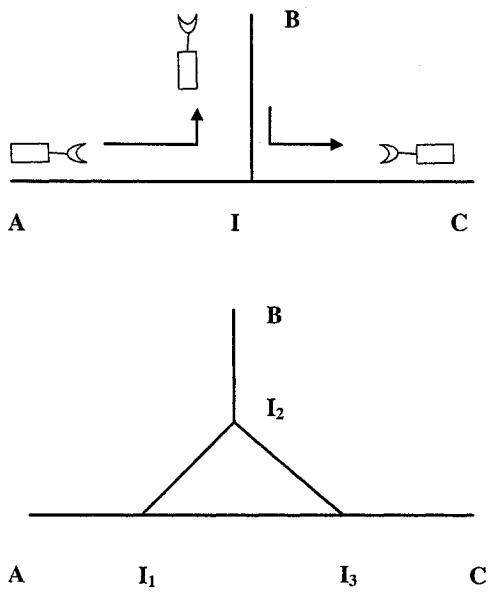

Figure 5.3 – changement d'orientation

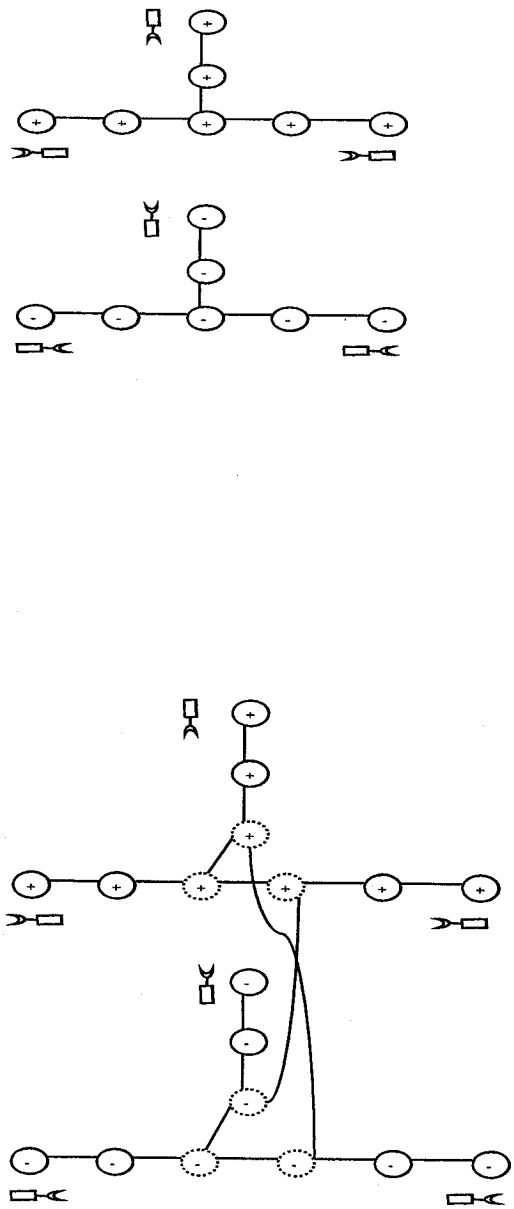

Figure 5.4 – description graphique du changement d'orientation

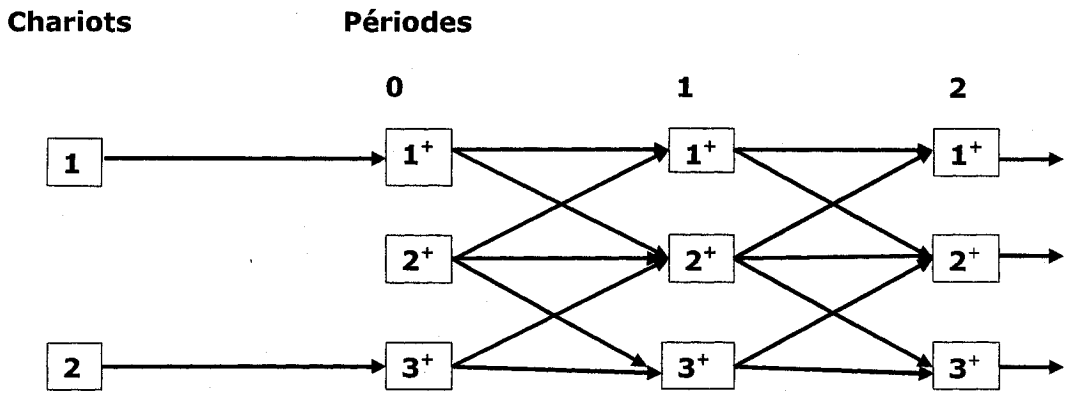

Figure 5.5 – graphe espace-temps utilisé (réseau +)

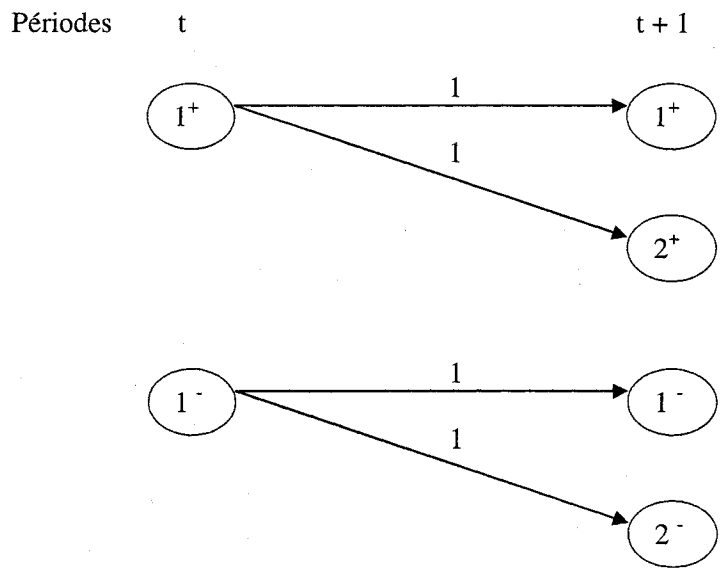

Figure 5.6 – arcs issus d'un noeud terminal

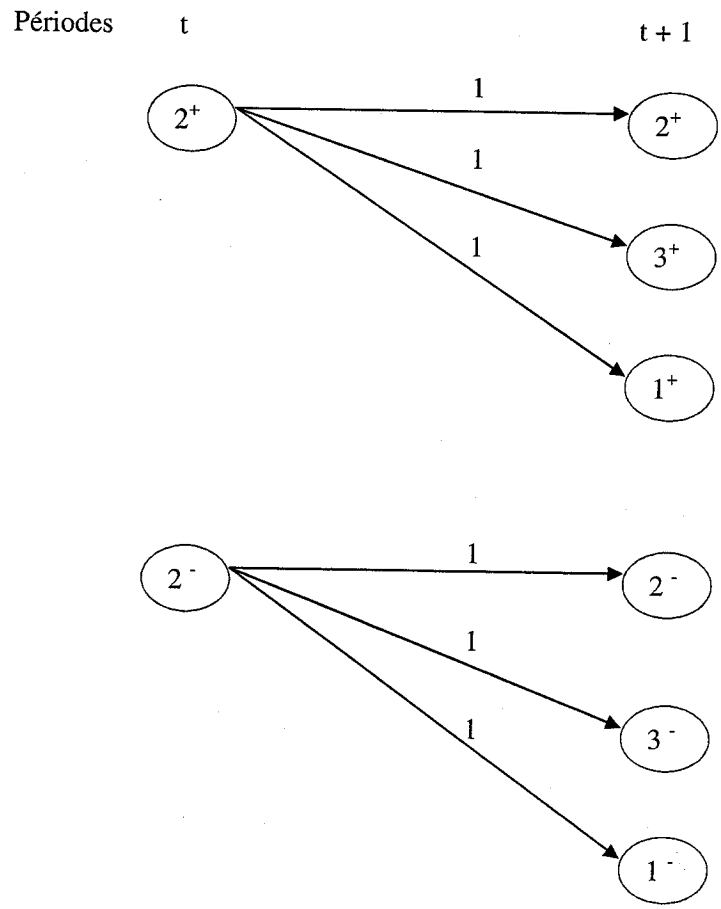

Figure 5.7 – arcs issus d'un noeud interne

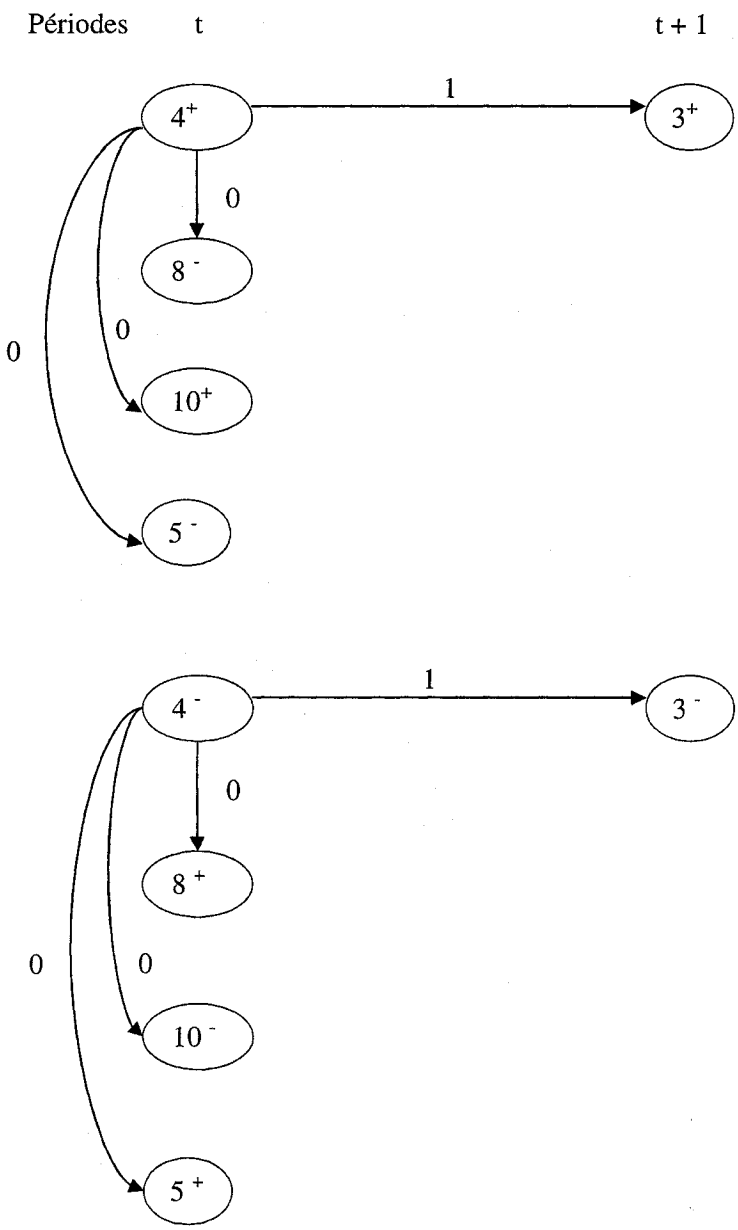

Figure 5.8 – arcs issus d'une intersection en carré

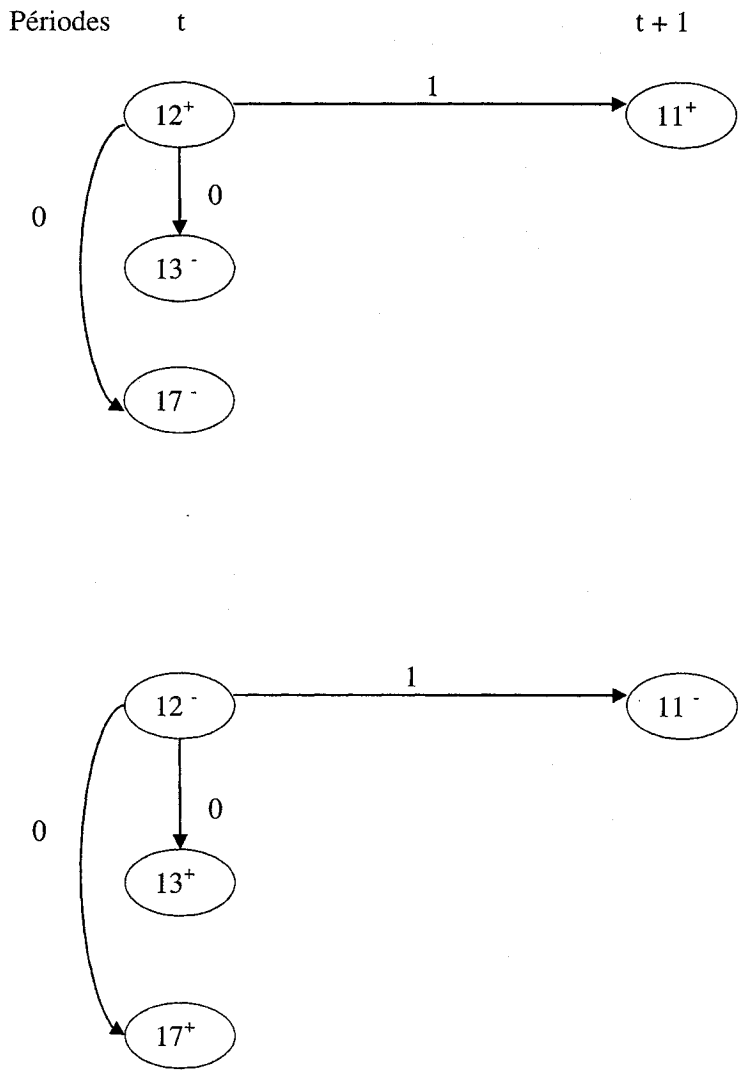

Figure 5.9 – arcs issus d'une intersection en triangle

Le modèle mathématique suivant est un modèle multiflux avec un type de flux par chariot. Le passage à certains noeuds du graphe espace-temps est fixé (pour l'exécution des tâches de chargement et déchargement). Tous les arcs de galerie sont de longueur 1 tandis que les arcs d'intersection sont de longueur 0. Les chariots peuvent partir de n'importe quel noeud à la période initiale (exception faite des points d'intersection). Les données nécessaires à la compréhension du modèle sont présentées ci-dessous. Certaines données sont générés automatiquement une seule fois au début de l'algorithme hybride pour créer le réseau physique.

Ensembles et paramètres du modèle de PLNE

U :	durée totale des tâches (importée du modèle de PC).
$t^c[\cdot]$:	temps de début de chargement (importés du modèle de PC).
$t^d[\cdot]$:	temps de début de déchargement (importés du modèle de PC).
v_r :	chariot affecté à la requête r (donnée importée du modèle de PC).
V :	ensemble des chariots.
n^v :	noeud de départ du chariot v.
R :	ensemble des requêtes de transport.
G :	ensemble des arêtes associées aux arcs de galerie.
N^g :	ensemble des noeuds de galerie.
n^+ :	nombre de noeuds étiquetés $+$.
N :	ensemble de tous les noeuds.
N_k^i :	ensemble des noeuds de l'intersection k.
A^i :	ensemble des arcs de toutes les intersections.
A_k^i :	ensemble des arcs de l'intersection k.
A :	ensemble de tous les arcs de galerie.
$A^f[i]$	ensemble de tous les arcs sortant du noeud i.
$A_g^f[i]$	ensemble de tous les arcs de galerie sortant du noeud i.

$A^f_{int}[i]$	ensemble de tous les arcs d'intersection sortant du noeud i.
$A^t[i]$	ensemble de tous les arcs entrant au noeud i.
$A^t_g[i]$	ensemble de tous les arcs de galerie entrant du noeud i.
$A^t_{int}[i]$	ensemble de tous les arcs d'intersection entrant du noeud i.
$A^w[i]$:	arc dont l'origine et la destination coincide avec le noeud i.
a	est un arc d'origine et de destination orientées +.
$p[a]$	arc de même direction que a avec origine et destination orientés -.
$o[a]$	noeud origine de l'arc a.
$d[a]$	noeud destination de l'arc a.
$n[r]$	noeud de chargement associé à la requête r.
$A^o_+[a]$:	opposé de l'arc a avec origine et destination de a orientés +.
$A^o_-[a]$:	opposé de l'arc a avec origine et destination de a orientés -.

Variables du modèle de PLNE :

$X^t_{k,a}$	variable booléenne = 1 si le chariot k débute le parcours de l'arc de galerie a à la période t et 0 sinon.
$Y^t_{k,a}$	variable booléenne = 1 si le chariot k débute le parcours de l'arc d'intersection a à la période t et 0 sinon.

Le modèle de PLNE est décrit ci-dessous. La fonction objectif du modèle de PLNE consiste en la minimisation des déplacements improductifs pour obtenir des déplacements cohérents et économiser de l'énergie. Toutefois, dans la mesure où une seule

solution réalisable à ce modèle est cherchée, la solution de routage trouvée essaie d'éviter le plus possible les déplacements improductifs.

$$Min \sum_{k\in V,\, t\in[1,..,(U-1)],\, a\in A:\, o[a]\neq d[a]} X^t_{k,a}$$

s.c.

$$\sum_{a\in A^f[n^k]} X^o_{k,a} = 1 \qquad \forall\, k \in V \tag{5.21}$$

$$\sum_{a\,\in\, A^t_g[i]} X^{t-1}_{k,a} + \sum_{a\,\in\, A^t_i[i]} Y^t_{k,a} = \sum_{a\,\in\, A^f_g[i]} X^t_{k,a} + \sum_{a\,\in\, A^f_i[i]} Y^t_{k,a} \tag{5.22}$$
$$\forall\, i \in N, k \in V, t \in [1,..,(U-1)]$$

$$\sum_{a\,\in\, A} X^t_{k,a} = 1 \qquad \forall\, t \in [0,..,(U-1)], k \in V \tag{5.23}$$

$$\sum_{k\,\in\, V, a\,\in\, A^f_g[i]} X^t_{k,a} \leq 1 \qquad \forall\, t \in [0,..,(U-1)], i \in N^g \tag{5.24}$$

$$\sum_{k\,\in\, V, a\,\in\, A^f_g[i]} X^t_{k,a} + \sum_{k\,\in\, V, a\,\in\, A^f_i[i]} Y^t_{k,a} \leq 1$$
$$\forall\, t \in [1,..,(U-1)], i \in N^i_k, k \in V \tag{5.25}$$

$$X^{t^c[r]}_{v_r, A^w[n[r]]} = 1 \qquad \forall\, r \in R \tag{5.26}$$

$$X^{t^c[r]+1}_{v_r, A^w[n[r]]} = 1 \qquad \forall\, r \in R \tag{5.27}$$

$$X^{t^d[r]}_{v_r, A^w[1]} = 1 \qquad \forall\, r \in R \tag{5.28}$$

$$\sum_{k\,\in\, V, a\,\in\, A^i_k} Y^t_{k,a} \leq 1 \qquad \forall\, t \in [1,..,(U-1)], k \in V \tag{5.29}$$

$$\sum_{k\,\in\, V, a\,\in\, Arcs: d[a]\,\in\, N^i_k} X^t_{k,a} \leq 1 \qquad \forall\, t \in [1,..,(U-1)], k \in V \tag{5.30}$$

$$\sum_{k\,\in\, V} (X^t_{k,u} + X^t_{k,A^o_+[u]} + X^t_{k,p[u]} + X^t_{k,A^o_-[p[u]]}) \leq 1$$
$$\forall\, t \in [0,..,(U-1)], u \in G : \big(o[u] + 1 \leq d[u] \leq n^+\big) \tag{5.31}$$

$$\sum_{k\,\in\, V, a\,\in\, A^t_g[i]} X^t_{k,a} \leq 1 \qquad \forall\, t \in [0,..,(U-1)], i \in N^g \tag{5.32}$$

Les contraintes (5.21) spécifient la position initiale de chaque chariot. Les contrain-tes (5.22) sont des contraintes de conservation de flux. Les contraintes (5.23) signifient que chaque chariot doit se trouver à un endroit unique à chaque période (et aussi un chariot ne peut être dans une intersection pendant deux périodes consécutives). Les contraintes (5.24) imposent la présence d'au plus un chariot sur chaque arc à chaque période. Les contraintes (5.25) assurent qu'il a au plus un chariot sur un noeud intersection à chaque période. Les contraintes (5.26) et (5.27) signifient que chaque chariot doit être à son point de chargement au bon moment puis y rester deux périodes pour effectuer son chargement de minerai. Les contraintes (5.28) imposent que les chariots qui doivent décharger du minerai doivent se présenter au bon endroit au bon moment et y passer une période. Les contraintes (5.29) assurent qu'au plus un chariot peut être présent sur chaque intersection à chaque période tandis que les contraintes (5.30) spécifient que deux chariots ne peuvent entrer en même temps dans une même intersection. Les contraintes (5.31) sont des contraintes d'évitement de collisions sur les arcs de galerie tandis que les contraintes (5.32) sont des contraintes d'évitement de collisions sur les noeuds de galerie (les noeuds sont numérotés).
Les contraintes (5.23) et (5.30) sont redondantes et peuvent être éliminées.

Stratégie de recherche (PLNE)

Dans cette partie, la stratégie de recherche utilisée dans l'arbre de branchement est essentiellement basée sur la fixation de variables liées à des événements précis (position initiale de chaque chariot, début des chargements et déchargements qui sont donnés). Pour toutes les autres variables non fixées, on branche d'abord à droite (variable binaire fixée à 1). Nous avons aussi utilisé l'algorithme du simplexe dual au noeud racine de l'arbre de recherche suivi de l'algorithme du simplexe réseau dans les autres noeuds tout en mettant l'emphase sur les solutions réalisables cachées (option de CPLEX 9.0).

5.2.6 Tests

Les réseaux utilisés pour les tests et les résultats obtenus sont présentés respectivement aux sections 5.2.6.1 et 5.2.6.2.

Réseaux et données utilisés

Nous avons utilisé trois réseaux qui peuvent être considérés comme des agrandissements successifs d'un réseau de galeries. Le troisième réseau possède les caractéristiques suivantes qui permettent de tester les limites aussi bien du modèle de PC que du modèle de PLNE :

- les galeries de chargement sont toutes de même longueur ce qui crée des symétries dans le modèle de PC ;

- les galeries sont allongées, ce qui augmente la taille du modèle de PLNE par rapport au modèle de PLNE des deux premiers réseaux (voir figure 5.1) ;

- il y a un nombre accru de chariots et de requêtes de transport à satisfaire par rapport aux deux premiers réseaux.

Le tableau 5.1 présente la description détaillée d'une problème pour chaque réseau. Aux extrémités de chaque réseau se trouvent le noeud de déchargement (toujours numéroté 1) et les autres noeuds de chargement. Sur chaque noeud de chargement, il y a un nombre de collectes de minerai à faire, nombre évalué à l'avance. On veut faire la planification pour au plus trois collectes par noeud de chargement. Il y a un délai minimal entre deux déchargements ou chargements consécutifs à respecter. En effet, lorsque deux chariots se suivent sur un même point de tâche (chargement/déchargement), le premier chariot doit avoir assez de temps pour exécuter sa tâche et sortir complèment de la galerie avant que le second chariot ne parcourt la galerie pour commencer sa tâche. La transition minimale entre deux déchargements

Tableau 5.1 – description détaillée de quelques problèmes

Réseau Utilisé	Numéro Problème	Noeud et Chargements	Nombre chariots	Noeud Initial
1	7	6 : 2 12 : 3 16 : 2	4	Veh1 : 12 Veh2 : 8 Veh3 : 6 Veh4 : 14 Veh5 : 11
2	4	7 : 2 9 : 3 16 : 3 20 : 2	5	Veh1 : 15 Veh2 : 38 Veh3 : 2 Veh4 : 6 Veh5 : 19
3	1	10 : 3 15 : 3 25 : 3 35 : 3 40 : 3	6	Veh1 : 12 Veh2 : 3 Veh3 : 37 Veh4 : 18 Veh5 : 7 Veh6 : 28

consécutifs est de cinq périodes tandis que celle entre deux chargements consécutifs est de 15 périodes. Chaque déchargement dure une période tandis que tout chargement dure deux périodes (le chargement est habituellement plus long que le déchargement). Nous supposons qu'aucun chariot ne peut se trouver initialement sur une intersection.

L'horizon choisi est variable : il dépend du nombre de chargements contenus dans chaque problème. Aussi pour chaque réseau, le nombre de chariots varie d'un problème à un autre. Le nombre total de chariots considéré dans chaque réseau, peut dépasser le nombre de noeuds de chargement d'une unité. Ces problèmes permettent d'analyser le cas où il existerait un chariot dédié à l'augmentation de la performance de la mine souterraine (ce chariot est surnommé flottant). Nous n'avons pas testé les problèmes à deux chariots car, dans ce cas, l'utilisation d'un système automatisé pour gérer la mine souterraine n'est pas pertinente.

Résultats obtenus

Les tests sont faits avec OPLScript 3.7 sur un ordinateur de type Pentium 4, 2.4 GHz, 512 Mo (RAM). Ilog Scheduler 6.0 est utilisé pour résoudre la partie PC (ordonnancement) et CPLEX 9.0 pour la partie PLNE (routage orienté sans conflits). 115 problèmes ont été générées aléatoirement et testées. Pour le réseau 1, treize problèmes avec un nombre de chariots variant de trois à quatre ont été testés. Pour le réseau 2, quinze problèmes avec un nombre de chariots variant de trois à cinq chariots tandis que pour le réseau 3, onze problèmes avec avec un nombre de chariots variant de trois à six chariots.

Les résultats obtenus sont affichés dans les tableaux 2 à 6. À chaque problème est associé un nombre total de chargements à faire à partir des points de chargement de minerai (voir colonne *Chargements* de chaque tableau de résultats). Précisons que deux problèmes peuvent avoir le même nombre de chargements à faire mais avec des répartitions de chargements et des points de départ de chariots différents. Pour chaque problème et pour un nombre de chariots fixé, le temps de calcul du modèle de PC (en secondes) est présenté dans la première colonne (TPC). Celui du modèle de PLNE se trouve sur la deuxième colonne ($TPLNE$). *Duree*, la durée totale de toutes les tâches (makespan), est affichée dans la troisième colonne tandis que $nbIter$, c'est-à-dire le nombre de fois qu'au moins un conflit a été détecté dans la partie routage, est affiché dans la quatrième colonne. Un trait (-) signifie qu'aucune solution au modèle de PC n'a été trouvée au bout de 20 minutes, tandis qu'un double trait signifie qu'aucune solution au modèle de PLNE n'a été trouvée au bout de 20 minutes.

Dans le réseau 1 (voir le tableau 5.2), 96 % des problèmes sont résolus, chacune en moins de 40 secondes. Le modèle de PC est ici particulièrement efficace car il anticipe bien sur les conflits et qu'à l'exception d'un, tous les problèmes ont été résolus en moins d'une seconde. Dans le problème 3 avec trois chariots, le modèle de PC a généré une solution avec conflits une seule fois. Le problème 3 avec quatre chariots n'a pas été résolu car la borne inférieure du makespan est trop éloignée de sa valeur optimale, ce qui a engendré beaucoup d'itérations. L'utilisation d'un chariot flottant (problèmes avec quatre chariots) ne crée pas de congestion supplémentaire significa-

tive dans la mine. En ce qui concerne le réseau 2 (voir les tableaux 5.3 et 5.4), tous les problèmes ont été résolus en moins d'une minute et demie. Le modèle de PC est assez efficace même s'il y a sept problèmes où des conflits n'ont pas été totalement anticipés. Il s'agit des problèmes 10 et 13 avec quatre chariots, des problèmes 9 - 10 - 11 - 12 - 13 avec cinq chariots. L'utilisation du chariot flottant engendre au passage une légère augmentation de la congestion dans la mine mais toutes les problèmes associés sont résolus. Dans le réseau 3 (voir les tableaux 5.5 et 5.6), 37 % des problèmes avec trois chariots n'ont pas été résolus. Les symétries générées par la nature spécifique du réseau 3 font qu'il est difficile de trouver un ordonnancement pour ces problèmes. Lorsque le nombre de chariots est augmenté à quatre, alors tous les problèmes sont résolus avec une anticipation complète des conflits dans la partie PC. Lorsque le nombre de chariots disponibles passe à cinq alors 72 % des problèmes sont résolus. En ce qui concerne les deux problèmes non résolus (problèmes 2 et 11 avec cinq chariots), un routage réalisable n'a pas été trouvé car on est dans la situation où une solution à la relaxation linéaire est trouvée sans qu'une solution entière ne soit disponible. Notons que le modèle de PC génère des solutions relativement vite. Il faut aussi mentionner que le modèle de PC n'a pas pu anticiper tous les conflits. Enfin, lorsque le chariot flottant est introduit (problèmes avec six chariots), le pourcentage de problèmes résolus est très bas (environ 27%). Ici le réseau est très congestionné (solution de routage très difficile à trouver). En outre, dans les problèmes résolus, tous les conflits n'ont pas été anticipés. Ainsi, dans le réseau 3, l'utilisation du chariot flottant devrait être évitée car son impact est négatif. Le temps de calcul est arbitrairement fixé à 20 minutes. Un temps de calcul plus élevé aurait pu nous permettre de trouver des solutions à des problèmes déclarés non résolus. Par exemple, le problème 3 avec trois chariots du réseau 3 serait classé résolu.

Les résultats obtenus en termes de durée totale de toutes les activités sont en partie inattendus, ce qui explique leur discussion séparée. Dans le réseau 1, l'augmentation du nombre de chariots disponibles n'a pas d'impact sur la durée totale de toutes les tâches. Puisque tous les problèmes sont résolus avec trois chariots, il n'est pas nécessaire de passer à quatre chariots. Ces résultats s'expliquent par la taille du réseau, les temps de transition minimaux et la qualité de la borne inférieure du modèle de PC. Dans le réseau 2, le passage de trois à quatre chariots disponibles a un impact limité

sur la durée totale de toutes les tâches à l'exception des problèmes 8 et 13 à quatre chariots où la durée totale de toutes les tâches augmente à cause de la congestion. L'ajout d'un chariot avec un certain point de départ et une orientation donnée peut causer du retard. Pour les problèmes 3, 4, 5 et 10, la durée totale des tâches baisse du fait de l'ajout d'un chariot proche d'un point de chargement. Lorsque le nombre de chariots disponibles passe à cinq, la durée totale des tâches augmente d'une période dans certains cas (problèmes 3 et 7). Pour le problème 8, le makespan augmente lorsqu'on passe de trois à quatre chariots puis avec cinq chariots, il y a un retour à une durée totale de tâches obtenue avec trois chariots. Ceci est dû à la proximité du point de départ du chariot ajouté d'un point de chargement. Dans le réseau 3, les résultats obtenus sont conformes à nos attentes. À l'exception du problème 10, la durée totale des tâches baisse lorsque le nombre de chariots disponibles passe de trois à quatre. Lorsque le nombre de chariots disponibles est égal à six, il n'y a pas de gain de temps d'exécution des tâches à l'exception du problème 2.

En résumé, lorsque le nombre de chariots disponibles augmente de un, c'est la position de départ et l'orientation du chariot supplémentaire qui ont un impact sur la durée totale de toutes les tâches. Cet impact peut être positif (proximité d'un point de chargement) ou négatif (retard engendré dû à une orientation inappropriée). D'où l'importance d'un bon choix des points d'attente (qui sont en fait des points de départ potentiels des chariots).

L'anticipation sur les conflits dans le modèle de PC (grâce à l'implantation de la notion de gel de galeries) a donné des résultats très satisfaisants. Toutefois, elle n'est pas complète car les galeries intermédiaires n'ont pas été prises en compte. Les conflits résiduels peuvent survenir de deux façons :

- soit il y a un conflit dans une galerie parce qu'un chariot inoccupé bloque le passage à un autre chariot qui doit aller exécuter sa tâche au bout de la galerie ; ce chariot inoccupé n'a pas assez de temps pour sortir de la galerie et céder la place (ceci se passe dans la partie PLNE alors que l'ordonnancement est fait dans la partie PC) ;

Tableau 5.2 – réseau 1 (3 et 4 chariots)

		3 chariots				4 chariots			
Problème	**Chargements**	**TPC**	**TPLNE**	**Duree**	**nbIter**	**TPC**	**TPLNE**	**Duree**	**nbIter**
1	9	0.05	2.63	75	0	0.03	7.88	75	0
2	7	0.03	2.95	59	0	0.16	7.05	59	0
3	6	0.02	1.92	51	1	-	-	-	-
4	7	0.02	2.36	59	0	0.02	3.31	59	0
5	7	0.03	2.53	57	0	0.01	5.78	57	0
6	5	0.01	1.44	43	0	0.02	3.02	43	0
7	6	0.03	1.78	51	0	0.03	4.41	51	0
8	9	0.08	2.99	75	0	0.03	12.57	75	0
9	7	0.08	2.17	59	0	0.14	2.87	59	0
10	6	0.01	2.23	53	0	0.03	35.46	53	0
11	7	0.02	2.08	59	0	0.02	2.99	59	0
12	7	0.05	1.91	59	0	0.05	5.84	59	0
13	5	0.02	1.37	43	0	0.01	2.31	43	0

Tableau 5.3 – réseau 2 (3 et 4 chariots)

		3 chariots				4 chariots			
Problème	**Chargements**	**TPC**	**TPLNE**	**Duree**	**nbIter**	**TPC**	**TPLNE**	**Duree**	**nbIter**
1	12	0.78	5.58	99	0	0.20	81.72	99	0
2	9	0.08	3.67	75	0	0.05	13.54	75	0
3	10	0.44	4.02	85	0	23.26	12.33	84	0
4	10	0.05	4.27	85	0	0.69	13.66	83	0
5	8	0.03	3.47	69	0	0.03	5.67	67	0
6	8	0.03	3.2	67	0	0.01	7.19	67	0
7	7	0.13	3.23	60	0	0.27	6.46	60	0
8	8	0.11	3.63	67	0	0.03	8.44	75	0
9	12	0.11	5.52	99	0	0.08	15.11	99	0
10	9	0.08	11.79	99	0	0.03	11.19	75	1
11	10	0.08	3.68	83	0	0.05	9.63	83	0
12	10	0.05	4.35	83	0	0.06	6.78	83	0
13	8	0.05	3.90	67	0	0.19	9.28	69	6
14	8	0.05	3.66	67	0	0.03	4.85	67	0
15	7	0.02	2.93	59	0	0.03	8.86	59	0

Tableau 5.4 – réseau 2 (5 chariots)

Problème	Chargements	TPC	TPLNE	Duree	nbIter
1	12	0.03	32.72	99	0
2	9	0.11	44.20	75	0
3	10	0.03	47.24	83	0
4	10	18.01	81.70	83	0
5	8	0.03	13.44	67	0
6	8	0.02	22.60	67	0
7	7	0.02	18.55	59	0
8	8	0.03	13.15	67	0
9	12	0.08	11.79	99	1
10	9	0.03	14.46	75	1
11	10	0.06	62.14	83	1
12	10	0.05	18.73	83	1
13	8	0.15	10.58	69	5
14	8	0.03	9.10	67	0
15	7	0.02	38.13	59	0

Tableau 5.5 – réseau 3 (3 et 4 chariots)

		3 chariots				4 chariots			
Problème	Chargements	TPC	TPLNE	Duree	nbIter	TPC	TPLNE	Duree	nbIter
1	15	-	-	-	-	3.35	102.62	156	0
2	13	251.92	346.66	147	0	0.14	79.63	135	0
3	11	-	-	-	-	0.22	91.46	115	0
4	11	15.43	30.63	127	0	0.11	73.39	115	0
5	10	93.20	32.44	113	6	0.23	221.54	104	0
6	10	-	-	-	-	0.19	50.20	109	0
7	10	9.79	24.94	117	0	0.27	6.46	109	0
8	15	-	-	-	-	8.49	96.48	155	0
9	11	25.88	25.10	128	0	0.17	47.72	115	0
10	11	3.69	20.75	112	0	0.61	57.27	115	0
11	10	14	21.98	117	0	4.62	61.41	109	0

Tableau 5.6 – réseau 3 (5 et 6 chariots)

		3 chariots				4 chariots			
Problème	Chargements	TPC	TPLNE	Duree	nbIter	TPC	TPLNE	Duree	nbIter
1	15	6.95	530.20	156	1	0.42	- -	- -	- -
2	13	0.094	- -	- -	- -	0.11	500.61	135	1
3	11	0.44	108.10	115	1	0.11	- -	–	- -
4	11	0.08	83.74	115	1	0.08	176.48	115	1
5	10	-	-	-	-	7.73	- -	- -	- -
6	10	0.25	468.34	109	1	9.79	- -	- -	–
7	10	0.44	334.52	109	1	4.92	- -	- -	- -
8	15	0.97	220.17	155	1	1.41	- -	- -	- -
9	11	0.09	114.78	115	1	0.08	163.64	115	1
10	11	0.92	224.59	115	1	0.11	- -	- -	- -
11	10	2.86	- -	- -	- -	20.93	- -	- -	- -

- soit il y a un conflit dans une galerie intermédiaire c'est-à-dire une galerie située entre deux intersections. Il s'agit, par exemple, de la galerie 10 - 11 - 12 du réseau 2. Un conflit peut survenir quand un chariot revient de la galerie de déchargement (voyage à vide pour aller effectuer un nouveau chargement) tandis qu'un autre chariot est déjà chargé et se dirige vers la galerie de déchargement ;

Pour éviter ces deux types de conflits, une solution tout à fait acceptable serait de créer dans chacune des galeries (en priorité les galeries intermédiaires) une ou deux petites voies latérales de dégagement (petites impasses) pour permettre à un chariot de céder la place à d'autres chariots.

Le modèle de PC est très efficace pour trois raisons principales : (1) l'analyse du problème permet de générer automatiquement des bornes inférieure et superieure très utiles pour l'aspect optimisation du modèle de PC. C'est la bonne qualité de la borne inférieure qui a plus d'impact sur le modèle de PC ; (2) la stratégie de recherche qui permet d'ordonner les activités sur les ressources unaires qu'elles doivent utiliser : (3) les contraintes globales dont les algorithmes de réduction de domaine sont très efficaces. Le modèle de PLNE a donné des résultats relativement bons vu sa grande taille, la difficulté d'éviter les conflits et de préciser l'orientation des chariots et le temps de résolution limité à vingt minutes.

5.2.7 Conclusion

Dans cet article, nous avons résolu un problème d'ordonnancement et routage intégrés d'une flotte de chariots dans une mine souterraine grâce une méthode de décomposition hybride qui combine la programmation par contraintes et la programmation linéaire en nombres entiers. Notre approche décompose le problème en deux parties : la première partie consiste en l'ordonnancement des tâches de chargement et déchargement de minerai tandis que la seconde partie résout le routage sans conflits des

chariots avec prise en compte de l'orientation des chariots. Les problèmes considérés contiennent jusqu'à six chariots. Un effort substantiel est fait dans l'anticipation des conflits (modèle de PC) et la cohérence de l'orientation des chariots (modèle de PLNE). Le traitement de l'orientation des chariots dans la littérature est souvent négligé ou fait de façon inexacte. Notre approche traite de façon exacte l'orientation des chariots. En outre, une analyse de scénarios d'agrandissements est faite. Notre méthode pourrait servir au design d'une mine. Notre algorithme constitue un bon outil d'aide au dimensionnement de la flotte de chariots et peut être utilisé pour rapidement replanifer le travail en fonction de modifications aux galeries (bris de chariots, chutes de roches, fuites d'eaux souterraines). Il serait intéressant de tester notre approche dans d'autres contextes où il y aurait plus de tâches et de chariots, de développer des stratégies d'évitement de solutions symétriques dans le modèle de PC, de considérer le cas où deux points de déchargement coexistent. Une mine contient souvent deux points de déchargement : l'un servant à décharger le minerai utile, l'autre les rebuts (dans cette application, la maximisation du plan de production de minerai utile est visée). Cela permettrait d'éprouver un peu plus une version modifiée du modèle de PC. L'introduction de deux points de déchargement permettrait aussi de résoudre des problèmes dans d'autres contextes (terminaux à conteneurs par exemple).

Références

APT, K. (2003). *Principles of Constraint Programming.* Cambridge University Press.

BAPTISTE, P., LE PAPE, et NUITJEN, W. (2001). *Constraint based Scheduling : Applying Constraint Programming to Scheduling Problems.* Operations Research/Management Science. Kluwer Academic Publishers, Dordrecht.

BEAULIEU, M., GAMACHE, C. (2004). An Enumeration Algorithm for Solving the Fleet Management Problem in Underground Mines. Cahiers du Gerad G-2004-52, HEC Montréal, Canada.

BIGRAS, L-P. et GAMACHE, M. (2002). A Solution Approach for Real-time Fleet Management System : An Application to Underground Mining. Cahiers du Gerad G-2002-66, HEC Montréal, Canada.

BIGRAS, L-P. et GAMACHE, M. (2005). Considering Displacement Modes in the Fleet Management Problem. *International Journal of Production Research* **43** 1171–1184.

CO, C.G. et TANCHOCO, J.M.A. (1991). A Review of Research on AGVs Vehicle Management. *Engineering Costs and Production Economics* **21** 35–42.

CORRÉA, A.I., LANGEVIN, A. et ROUSSEAU, L-M. (2004). Dispatching and Conflict-Free Routing of Automated Guided Vehicles : A Hybrid Approach Combining Constraint Programming and Mixed Integer Programming. Régin, J-C. and Rueher, M., eds. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. First International Conference, CPAIOR. Lecture Notes in Computer Science. LNCS 3011.* Springer.

CORRÉA, A.I., LANGEVIN, A. et ROUSSEAU, L-M. (2005a). Scheduling and Routing of Automated Guided Vehicles : a Hybrid Approach. *Computers and Operations Research* To appear.

DESAULNIERS,G., LANGEVIN, A., RIOPEL, D. et VILLENEUVE, B. (2003). Dispatching and Conflict-Free Routing of Automated Guided Vehicles : An Exact Approach. *The International Journal of Flexible Manufacturing Systems* **15** 309–331.

DINCBAS, M., VAN HENTENRYCK, P.et SIMONIS, H. (1990). Solving Large Combinatorial Problems in Logic Programming. Journal of Logic Programming **8** 75–93.

FUJI, S., SANDOK, H. et HOHZAKI, R. (1988). Routing Control of Automated Guided Vehicles in FMS. *Proceedings of the USA-Japan Symposium on Flexible Automation. Minneapolis, MN.* 130–136.

GAMACHE, M., GRIMARD, R. et COHEN, P. (2004). A Shortest-Path Algorithm for Solving the Fleet Management Problem in Underground Mines. *European Journal of Operational Research* **166** 497–506.

GANESHARAJAH, T., HALL, N.G., et SRISKANDARAJAH, C. (1998). Design and Operational Issues in AGV-Served Manufacturing Systems. *Annals of Operations Research* **76** 109–154.

HOOKER, J. N. (2000). Logic-Based Methods for Optimisation. Wiley, New York.

HOOKER, J. N. et OTTOSSON, G. (2003). Logic-Based Benders Decomposition. *Mathematical Programming* **96** 33–61.

ILOG OPL STUDIO (2004). *Ilog OPL Studio 3.7* Language Manual.

JAFFAR, J. et MAHER, M. J. (1994). Constraint Logic Programming : A Survey. *Journal of Logic Programming* **19/20** 503–581.

KING, R.E. et WILSON, C. (1991). A Review of Automated Guided Vehicle System Design and Scheduling. *Production Planning and Control* **2** 44–51.

KRISHNAMURTHY, N.N., BATTA, R. et KARWAN, M.H. (1993). Developing Conflict-Free Routes for Automated Guided Vehicles. *Operations Research* **4** 1077–1090.

LANGEVIN, A., LAUZON, D., et RIOPEL, D. (1996). Dispatching, Routing and Scheduling of Two Automated Guided Vehicles in a Flexible Manufacturing System. *International Journal of Flexible Manufacturing Systems* **8** 246–262.

MARIOTT, K. et STUCKEY, P.J. (1998). Programming with Constraints. The MIT Press.

MILANO, M. (2004). Constraint and Integer Programming : Toward a Unified Methodology. Ramesh Sharda and Stefan Vob, eds. *Operations Research/Computer Science Interfaces Series.* Kluwer Academic Publishers. 33-53.

QIU, L., HSU, W.-J. et WANG, H. (2002). Scheduling and Routing Algorithms for AGVs : A Survey. *International Journal of Production Research* **40** 745–760.

VAGENAS, N. (1991). Dispatch Control of a Fleet of Remote-Controlled/Automatic Load-Haul-Dump Vehicles in Underground Mines. *International Journal of Production Research.* **29** 2347–2363.

VAN HENTENRYCK, P. (1989). Constraint Satisfaction in Logic Programming. MIT Press, Cambridge, Mass.

WALLACE, M. (1996). Practical Applications of Constraint Programming. *Constraints.* **1** 139–168.

CHAPITRE 6 : CONCLUSION

Dans cette thèse, nous nous sommes attaqués à trois problèmes intégrés d'ordonnancement et routage sans conflits de chariots dans deux contextes : un atelier flexible et une mine souterraine. Les deux problèmes (première et deuxième méthodes de décomposition) étudiés dans le contexte de l'atelier flexible sont similaires. Nous avons présenté des algorithmes de décomposition hybrides qui combinent la PC et la PLNE. Ces algorithmes sont des variantes d'une méthode qui décompose les problèmes étudiés en deux parties : la première consiste en l'ordonnancement des tâches tandis que la seconde s'occupe du routage sans conflits.

Dans la première méthode de décomposition (qui est le point de départ des développements algorithmiques contenus dans les deux articles subséquents), le modèle de PC générait trop de solutions alternatives qui étaient irréalisables dans la partie PLNE. En outre, l'algorithme conçu n'était pas spécialisé au contexte de l'atelier flexible. Dans la deuxième méthode de décomposition, des coupes logiques de réalisabilité (de type *nogood*) sont implantées pour réparer les conflits et écarter les solutions optimales pour le modèle de PC mais irréalisables pour le modèle de PLNE. Dans la troisième méthode de décomposition, l'algorithme conçu est spécifique au type de réseau. La prévention partielle des conflits ainsi que l'orientation exacte des chariots y sont traitées. Nous nous sommes basés sur une analyse du problème pour prévenir les conflits et nous avons obtenu un modèle de PC qui fait un traitement presque complet des conflits. Les zones de conflits résiduels sont identifiés et des solutions pratiques sont proposées pour éviter la plupart de ces conflits résiduels. En outre, une analyse prospective de l'évolution de la mine est faite. L'algorithme développé dans la troisième méthode de décomposition pourrait être utilisé pour réaffecter rapidement les chariots aux requêtes suite à panne qui bloque toute une galerie (bris de chariots, fuite d'eau souterraine, chutes de roches).

D'après la classification de Qiu *et al.* (2002), les algorithmes sur les chariots automatiques peuvent être classés en trois grands groupes :

- les algorithmes d'optimisation pour des réseaux quelconques.

- les algorithmes pour la conception des réseaux.

- les algorithmes d'optimisation pour les réseaux à topologie spécifique.

Les algorithmes présentés dans les première et deuxième méthodes de décomposition peuvent être classés dans le premier groupe. L'algorithme présenté dans la troisième méthode de décomposition appartient au troisième groupe.

Après la présentation des trois algorithmes, nous pouvons dire que nous avons conçu une méthode de décomposition hybride qui consiste à résoudre d'abord une relaxation du modèle original (avec le modèle de PC). La partie relaxée est l'ensemble des contraintes de routage sans conflits. Puis on traite les contraintes de routage sans conflits. Le choix de la partie relaxée provient du fait qu'il est d'abord nécessaire de savoir quel chariot exécute quelles requêtes et à quels moments (la période d'exécution des tâches a un grand impact sur l'apparition de conflits).

Quatre objectifs de cette thèse étaient : (1) Intégrer deux niveaux d'optimisation (ordonnancement et routage sans conflits) dans des problèmes complexes de logistique. (2) Explorer les méthodes de décomposition hybrides de la PC et la PLNE avec comme objectif l'intégration de leurs forces. (3) Concevoir des méthodes fiables qui prennent en compte des contraintes pratiques et traitent des aspects souvent négligés dans la littérature comme l'orientation exacte des chariots. (4) Résoudre par une méthode exacte des problèmes intégrés d'ordonnancement et routage sans conflits contenant jusqu'à six chariots.

Les travaux effectués dans le contexte de la mine souterraine montrent l'efficacité des algorithmes conçus pour des réseaux spécifiques, tandis que, dans le contexte de l'atelier flexible, la méthode est moins efficace parce qu'elle s'appuie sur un algorithme conçu pour des réseaux quelconques. Lorsque le contexte le permet (comme dans le cas de la mine souterraine), quelques décisions sur l'implantation permettent d'éviter

la majeure partie des conflits résiduels. Dans le contexte de la mine souterraine, il s'agit de creuser deux ou trois fentes sur les parois des galeries intermédiaires pour permettre aux chariots de s'abriter et laisser passer d'autres chariots venant en sens inverse.

Les deux contextes étudiés dans cette thèse ne justifient pas l'utilisation d'un plus grand nombre de chariots (six) et requêtes de transport. Il serait intéressant de tester notre approche dans des contextes où il y aurait plus de chariots et de requêtes de transport, car souvent les mines souterraines contiennent deux points de déchargement même si le deuxièment point de déchargement sert surtout au transport de personnel et à l'évacuation de rebuts.

Les résultats obtenus suggèrent l'extension de notre méthode aux problèmes des gestion de terminaux à conteneurs où il y a un grand nombre de chariots et de tâches. Une autre voie de recherches futures est la prise en compte de deux points de déchargement dans le contexte de la mine souterraine. Les résultats obtenus nous encouragent à poursuivre des recherches dans l'intégration de plusieurs niveaux d'optimisation (par exemple, les problèmes intégrés d'entreposage et de tournées de véhicules). Une exploration des méthodes hybrides de la PC et la PLNE dans le domaine du transport multimodal est une avenue de recherches à considérer.

RÉFÉRENCES BIBLIOGRAPHIQUES

AJILI, F. et EL SAKKOUT, H. (2003). *A Probe-Based Algorithm for Piecewise Linear Optimization in Scheduling.* Annals of Operations Research, **118** 35–48.

APT, K. (2003). *Principles of Constraint Programming.* Cambridge University Press.

BAPTISTE, P., LE PAPE, et NUITJEN, W. (2001). *Constraint based Scheduling : Applying Constraint Programming to Scheduling Problems.* Operations Research/Management Science. Kluwer Academic Publishers, Dordrecht.

BEAULIEU, M., GAMACHE, C. (2004). An Enumeration Algorithm for Solving the Fleet Management Problem in Underground Mines. Cahiers du Gerad G-2004-52, HEC Montréal, Canada.

BECK, J.C. et REFALO, P. (2003). *A Hybrid Approach to Scheduling with Earliness and Tardiness Costs.* Annals of Operations Research, **118** 49–71.

BELDICEANU, N. et PETIT, T. (2004). Cost Evaluation of Soft Global Constraints. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. First International Conference, CPAIOR. Lecture Notes in Computer Science. LNCS 3011.* Springer. Nice, France.

BENOIST, T., LABURTHE, F. et ROTTEMBOURG, B. (2001). Lagrange Relaxation and Constraint Programming Collaborative Schemes for Travelling Tournament Problems.
Programm and papers presented at CPAIOR'01 : www.icparc.ic.ac.uk/cpAIOR01/.

BENOIST, T., GAUDIN, E. et ROTTEMBOURG, B. (2002). Constraint Programming Contribution to Benders Decomposition : A Case Study. *LNCS 2470. 8th International Conference, CP 2002 2470. 09/2002. Pascal Van Hentenryck, ed.*. Springer.

BIGRAS, L-P. et GAMACHE, M. (2002). A Solution Approach for Real-time Fleet Management System : An Application to Underground Mining. Cahiers du Gerad G-2002-66, HEC Montréal, Canada.

BIGRAS, L-P. et GAMACHE, M. (2005). Considering Displacement Modes in the Fleet Management Problem. *International Journal of Production Research* **43** 1171–1184.

BOCKMAYR, A. et KASPER, T. (1998). Branch and infer : A Unifying Framework for Integer and Finite Domain Constraint Programming. INFORMS *Journal on Computing* **10** 287–300.

CASEAU, Y. et LABURTHE, F. (1999). Heuristics for Large Constrained Routing Problems. *Journal of Heuristics* **5** 281–303.

CASEAU, Y., LABURTHE, F., LE PAPE, C. et ROTTEMBOURG, B. (2001). Combining Local and Global Search in a Constraint Programming Environment. *Knowledge Engineering Review* **16(1)** 41–68.

CHU, Y. et XIA, Q.(2004). Generating Benders Cuts for a General Class of Integer Programming Problems. Régin, J-C. and Rueher, M., eds. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. First International Conference, CPAIOR. Lecture Notes in Computer Science. LNCS 3011.* Springer. Nice, France.

CO, C.G. et TANCHOCO, J.M.A. (1991). A Review of Research on AGVs Vehicle Management. *Engineering Costs and Production Economics* **21** 35–42.

CORRÉA, A.I., LANGEVIN, A. et ROUSSEAU, L-M. (2004). Dispatching and Conflict-Free Routing of Automated Guided Vehicles : A Hybrid Approach Combining Constraint Programming and Mixed Integer Programming. Régin, J-C. and Rueher, M., eds. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. First International Conference, CPAIOR. Lecture Notes in Computer Science. LNCS 3011.* Springer.

CORRÉA, A.I., LANGEVIN, A. et ROUSSEAU, L-M. (2005a). Scheduling and Routing of Automated Guided Vehicles : a Hybrid Approach. *Computers and Operations Research* To appear.

CORRÉA, A.I., LANGEVIN, A. et ROUSSEAU, L-M. (2005b). Ordonnancement et Routage Intégré d'une Flotte de Chariots dans une Mine Souterraine. Cahiers du Gerad G-2005-58, HEC Montréal, Canada. Soumis pour publication à INFOR.

CSPLIB. *Constraint Programming Benchmark Library.* www.csplib.org.

DE BACKER, B., FURNON, V., SHAW, P., KILBY, P. et PROSSER, P. (2000). Solving Vehicle Routing Problems Using Constraint Programming and Metaheuristics. *Journal of Heuristics* **6** 481–500.

DEMASSEY, S., PESANT, G. et ROUSSEAU, L-M. (2005). Constraint Programming Based Column Generation for Employee Timetabling. Barták, R. and Milano, M., eds. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. Second International Conference, CPAIOR. Lecture Notes in Computer Science. LNCS 3524.* Springer.

DESAULNIERS,G., LANGEVIN, A., RIOPEL, D. et VILLENEUVE, B. (2003). Dispatching and Conflict-Free Routing of Automated Guided Vehicles : An Exact Approach. *The International Journal of Flexible Manufacturing Systems* **15** 309–331.

DINCBAS, M., VAN HENTENRYCK, P.et SIMONIS, H. (1990). Solving Large Combinatorial Problems in Logic Programming. Journal of Logic Programming **8** 75–93.

DROLET, M. (1991). Ordonnancement d'un Carnet de Commandes pour un Atelier Flexible de Sous-Traitance. Projet de Fin d'Études, 1991. Département de Génie Industriel, Ecole Polytechnique de Montréal, Canada.

EASTON, K., NEMHAUSER, G.L. et TRICK, M.A. (2002). Solving the Traveling Tournament Problem : A Combined Integer Programming and Constraint Programming Approach. *Proceedings of International Conference on the Practice and Theory of Automated Timetabling, PATAT'02.*

EL SAKKOUT, R. et WALLACE, M. G. (2000). Probe Backtrack Search for Minimal Perturbation in Dynamic Scheduling. *Constraints* **5(4)** 359–388. Special issue on Industrial Constraint-Directed Scheduling.

EREMIN, A. et WALLACE, M. (2001). Hybrid Benders Decomposition Algorithms in Constraint Logic Programming. *Lecture Notes in Computer Science. CP 2001. LNCS 2239.*

FAHLE, T. et SELLMANN, M. (2000). Constraint Programming Based Column Generation with Knapsack Subproblems. *Proceedings of International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, CP - AI - OR '00.* Paderborn, Germany. 33–44

FAHLE, T., JUNKER, U., KARISCH, S.E., KOHL, N., SELLMANN, M. et VAABEN, B. (2000). Constraint Programming Based Column Generation for Crew Assignment. Technical Report tr-ri-99-212. University of Paderborn.

FOCACCI, F., LODI, A. et MILANO, M. (1999a). Cost-Based Domain Filtering. *Proceedings of International Conference on Principles and Practice of Constraint Programming, CP99.* 189–203. Alexandria, VA, USA.

FOCACCI, F., LODI, A. et MILANO, M. (1999b). Integration of CP and OR Methods for Matching Problems. *Proceedings of International Workshop on the Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming for Combinatorial Optimization, CP-AI-OR 99,* Ferrara, Italy.

FRÜHWIRTH, T. et ABDENNADHER, S. (2003). Essentials of Constraint Programming. Springer Verlag.

FUJI, S., SANDOK, H. et HOHZAKI, R. (1988). Routing Control of Automated Guided Vehicles in FMS. *Proceedings of the USA-Japan Symposium on Flexible Automation. Minneapolis, MN.* 130–136.

GAMACHE, M., GRIMARD, R. et COHEN, P. (2004). A Shortest-Path Algorithm for Solving the Fleet Management Problem in Underground Mines. *European Journal of Operational Research* **166** 497–506.

GANESHARAJAH, T., HALL, N.G., et SRISKANDARAJAH, C. (1998). Design and Operational Issues in AGV-Served Manufacturing Systems. *Annals of Operations Research* **76** 109–154.

GENDRON, B., LEBBAH, H. et PESANT, G. (2005). Improving the Cooperation Between the Master Problem and the Subproblem in Constraint Programming Based Column Generation. Barták, R. and Milano, M., eds. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. Second International Conference, CPAIOR. Lecture Notes in Computer Science. LNCS 3524*. Springer.

GROUP (2002). *Introduction to ECLiPSe.*

HAJIAN, M., RODOSEK, R. et RICHARDS, B. (1995). Towards a Closer Integration for Finite Domain Propagation and Simplex Based Algorithms. Technical report. IC-PARC.

HOOKER, J. N. et OSORIO, G. (1999). Mixed Logical/Linear Programming. *Discrete Applied Mathematics* **96-97** 395–442.

HOOKER, J. N. (2000). Logic-Based Methods for Optimisation. Wiley, New York.

HOOKER, J. N. (2003). A Framework for Integrating Solution Methods. Bhargava, H.K. et Mong Ye, eds. *Computational modeling and Problem Solving in the Networked World* (Proceedings of ICS2003). Kluwer 3–30.

HOOKER, J. N. et OTTOSSON, G. (2003). Logic-Based Benders Decomposition. *Mathematical Programming* **96** 33–61.

HOOKER, J. N. (2004). A Hybrid Method for Planning and Scheduling. *ed.* Wallace, M., *CP 2004*. Springer-Verlag Berlin Heidelberg. LNCS 3258.

HOOKER, J. N. (2005). A Search-Infer-and-Relax Framework for Integrating Solution Methods. Barták, R. and Milano, M., eds. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. Second International Conference, CPAIOR. Lecture Notes in Computer Science. LNCS 3524.* Springer.

ILOG OPL STUDIO (2002). *Ilog OPL Studio 3.6* Language Manual.

ILOG (2003). *Ilog Concert technology.*
www.ilog.com/products/optimization/tech/concert.cfm.

ILOG OPL STUDIO (2004). *Ilog OPL Studio 3.7* Language Manual.

JAFFAR, J. et MAHER, M. J. (1994). Constraint Logic Programming : A Survey. *Journal of Logic Programming* **19/20** 503–581.

JAIN, V. et GROSSMANN, I.(2001). Algorithms for Hybrid MILP / CP Models for a Class of Optimization Problems. *Informs Journal on Computing* **13** 258–276.

JUNKER, U., KARISCH, S.E., KOHL, N., VAABEN, B., FAHLE, T. et SELLMANN, M. (1999). A Framework Constraint Programming Based Column Generation. *Proceedings of Fifth International Conference on Principles and Practice of Constraint Programming, CP99.* LNCS 1713, 261–274. Springer.

KING, R.E. et WILSON, C. (1991). A Review of Automated Guided Vehicle System Design and Scheduling. *Production Planning and Control* **2** 44–51.

KRISHNAMURTHY, N.N., BATTA, R. et KARWAN, M.H. (1993). Developing Conflict-Free Routes for Automated Guided Vehicles. *Operations Research* **4** 1077–1090.

LANGEVIN, A., LAUZON, D., et RIOPEL, D. (1996). Dispatching, Routing and Scheduling of Two Automated Guided Vehicles in a Flexible Manufacturing System. *International Journal of Flexible Manufacturing Systems* **8** 246–262.

LEE, J.H., LEE, B.H. et CHOI, M.H. (1998). A Real-Time Traffic Control Scheme of Multiple AGV Systems for Collision Free Minimum Time Motion : A Routing Table Approach. *IEEE Transactions on Systems, Man and Cybernetics-Part A : Systems and Humans.* **28** 347–358.

MARAVELIAS, C. T. et GROSSMANN, I. E. (2004). Using MILP and CP for the Scheduling of Batch Chemical Processes. Régin, J-C. and Rueher, M., eds. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. First International Conference, CPAIOR. Lecture Notes in Computer Science. LNCS 3011.* Springer. Nice, France.

MARIOTT, K. et STUCKEY, P.J. (1998). Programming with Constraints. The MIT Press.

MELHORN, K. et THIEL, S. (2000). Faster Algorithms for Bound-Consistency of the Sortedness and the Alldifferent Constraint. *Proceedings of International Conference on Principles and Practice of Constraint Programming, CP00.* 306–319. Singapore.

MICHEL, L., et VAN HENTENRYCK, P. (2000). Localizer. *Constraints.* **5** 41–81.

MILANO, M. (2004). Constraint and Integer Programming : Toward a Unified Methodology. Ramesh Sharda and Stefan Vob, eds. *Operations Research /Computer Science Interfaces Series.* Kluwer Academic Publishers. 33-53.

MOSEL (2004). www.dashoptimization.com/home/downloads/pdf/mosel.pdf.

OBOTH, C., BATTA, R. et KARWAN, M. (1999). Dynamic Conflict-Free Routing of Automated Guided Vehicles. *International Journal of Production Research* **37** 2003–2030.

PESANT, G. et GENDREAU, M. (1996). A View of Local Search in Constraint Programming. Freuder, E.C., editor. *Proceedings of the Second International Conference on Principles and Practice of Constraint Programming, CP'96.* LNCS 1118, 353–366. Springer-Verlag

PESANT, G., GENDREAU, M. et ROUSSEAU, J.M. (1997). GENIUS-CP : A Generic Single-Vehicle Routing Algorithm. Smolka, G., editor. *Principles and Practice of Constraint Programming, CP'97.* LNCS 1330, 420–433. Springer-Verlag

PESANT, G. et GENDREAU, M. (1999). A Constraint Programming Framework for Local Search Methods. Freuder, E.C., editor. *Journal of Heuristics* **5** 255–279.

PESANT, G. (2001). A Filtering Algorithm for Stretch Constraint. *Proceedings of International Conference on Principles and Practice of Constraint Programming, CP01.* 183–195. Pathos, Cyprus.

PESANT, G. (2004). A Regular Membership Constraint for Finite Sequence of Variables. *Proceedings of Tenth International Conference on Principles and Practice of Constraint Programming, CP'04.* LNCS 3258 482–495.

QIU, L., HSU, W.-J. et WANG, H. (2002). Scheduling and Routing Algorithms for AGVs : A Survey. *International Journal of Production Research* **40** 745–760.

RAJOTIA, S., SHANKER, K. et BATRA, J.L.(1998). A Semi-Dynamic Window Constrained Routing Strategy in an AGV System. *International Journal of Production Research* **36** 35–50.

REFALO, F. (1999). Tight Cooperation and its Application in Piecewise Linear Optimization. *Proceedings of Fifth International Conference on Principles and Practice of Constraint Programming, CP99.* LNCS 1713, 375–389. Springer.

REFALO, F. (2000). Linear Formulation for Constraint Programming Models and Hybrid Solvers. *Proceedings of Sixth International Conference on Principles and Practice of Constraint Programming, CP00.* 369–3383. Singapore.

RÉGIN, J. C. (1996). Generalized Arc Consistency for Global Cardinality Constraint. *Proceedings of AAAI/IAAI.* **1** 209–215. AAAI Press/The MIT Press.

RÉGIN, J. C. (1997). The Global Minimum Distance Constraint. Technical report. ILOG.

RÉGIN, J. C. (1999a). Arc Consistency for Global Cardinality Constraint with Costs. *Proceedings of International Conference on Principles and Practice of Constraint Programming, CP99.* 390–404. Alexandria, VA, USA.

RÉGIN, J. C. (1999b). The Symmetric Alldiff Constraint. *Proceedings of International Joint Conference in Artificial Intelligence, IJCAI'99.* 425–429. Stockholm, Sweden.

RÉGIN, J. C. (2002). Cost-Based Arc Consistency for Global Cardinality Constraint. *Constraints, an International Journal.* **7(3-4)** 387–405.

RÉGIN, J. C. (2005). Combination of Among and Cardinality Constraints. Barták, R. and Milano, M., eds. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. Second International Conference, CPAIOR. Lecture Notes in Computer Science. LNCS 3524.* Springer.

RODOSEK, R., WALLACE, M. G. AND HAJIAN, M.T. (1999). A New Approach to Integrating Mixed Integer Programming and Constraint Logic Programming. *Annals of Operations Research* **86** 63–87. Special issue on Advances in Combinatorial Optimization.

ROUSSEAU, L-M., GENDREAU, M. et PESANT, G.(2002a). A General Approach to the Physician Rostering Problem. *Annals of Operations Research.* **115** 193–205.

ROUSSEAU, L-M., GENDREAU, M. et PESANT, G.(2002b). Solving Small VRPTWs with Constraint Programming Based Column Generation. Régin, J-C. and Rueher, M., eds. *Integration of International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, CP-AI-OR'02.* Le Croisic, France.

ROUSSEAU, L-M.(2004). Stabilization Issues for Constraint Programming Based Column Generation. Régin, J-C. and Rueher, M. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. First International Conference, CPAIOR. Lecture Notes in Computer Science. LNCS 3011.* Springer. Nice, France.

SARASWAT, V. et VAN HENTENRYCK, P. (1995). Principles and Practice of Constraint Programming. 1995. The MIT Press.

SAVELSBERGH, M.W.P. (1997). A Branch-and-Price Algorithm for the Generalized Assignment Problem. *Operations Research* **45** 831–841.

SELLMANN, M. ZERVOUDAKIS, K., STAMATOPOLOUS ET FAHLE, T. (2000). Integrating Direct CP Search and CP-Based Column Generation for the Airline Crew Assignment problem. *Proceedings of International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, CP - AI - OR '00.* Paderborn, Germany.

SELLMANN, M. et FAHLE, T. (2003). Constraint Programming Based Lagrangian Relaxation for the Automatic Recording Problem. *Annals of Operations Research*, **118** 17–33.

THORSTEINSSON, E. S. (2001a). Branch-and-Check : A Hybrid Framework Integrating Mixed Programming and Constraint Logic Programming. *Principles and Practice of Constraint Programming - CP 2001. Lecture Notes in Computer Science 2239.* 16–30.

THORSTEINSSON, E. S. (2001b). Hybrid Approaches to Combinatorial Optimisation. PhD thesis, Carnegie Mellon University.

THORSTEINSSON, E. S.et OTTOSSON, G. (2001) Linear Relaxations and Reduced-Cost Based Propagation of Continuous Variable Subscripts. *Annals of Operations Research* **115** 15–29.

VAGENAS, N. (1991). Dispatch Control of a Fleet of Remote-Controlled/Automatic Load-Haul-Dump Vehicles in Underground Mines. *International Journal of Production Research.* **29** 2347–2363.

VAN HENTENRYCK, P. (1989). Constraint Satisfaction in Logic Programming. MIT Press, Cambridge, Mass.

VAN HENTENRYCK, P., et MICHEL, L. (2004). Scheduling Abstractions for Local Search. Régin, J-C. and Rueher, M., eds. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. First International Conference, CPAIOR. Lecture Notes in Computer Science. LNCS 3011.* Springer. Nice, France.

VAN HOEVE, W.J., PESANT, G. et ROUSSEAU, L-M. (2005) On Global Warming : Flow-Based Soft Global Constraint. *Journal of Heuristics* To appear.

WALLACE, M. (1996). Practical Applications of Constraint Programming. *Constraints* 1 139–168.

XPRESS-MP (2001). *Optimization, Dash* www.dash.co.uk.

ZHOU, J. (1996). A Constraint Program for Solving the Job-Shop Problem. *Proceedings of International Conference on Principles and Practice of Constraint Programming, CP96.* 510–524. Cambridge.

ZHOU, J. (1997). Computing Smallest Cartesian Product of Intervals : Application to the Jobshop Scheduling Problem. PhD thesis, Université de la Méditerranée, Marseille.

APPENDIX A : DISPATCHING AND CONFLICT FREE ROUTING OF AUTOMATED GUIDED VEHICLES: A HYBRID APPROACH COMBINING CONSTRAINT PROGRAMMING AND MIXED INTEGER PROGRAMMING

Ayoub Insa Corréa, André Langevin and
Louis-Martin Rousseau,

Département de mathématiques et de génie industriel, Ecole Polytechnique de Montréal, C.P. 6079, Succ. Centre-ville, Montréal (Québec) Canada H3C 3A7

This paper reports on the on-going development of a hybrid approach for dispatching and conflict free routing of automated guided vehicles for material handling in manufacturing. The approach combines Constraint Programming for scheduling and Mixed Integer Programming for routing without conflict. The objective of this work is to provide a reliable method for solving instances with a large number of vehicles. The proposed approach can also be used heuristically to obtain very good solution quickly.

Key words: Automated guided vehicles, hybrid model, Constraint programming, material handling systems, routing.

A.1. Introduction

This study focuses on automated guided vehicles (ASGVs) in a flexible manufacturing system (FMS). An AGV is a material handling equipment that travels on a network of

paths. The FMS is composed of various cells, also called working stations, each with a specific function such as milling, washing, or assembly. Each cell is connected to the guide path network by a pick-up / delivery (P/D) station where the pick-ups and deliveries are made. The guide path is composed of aisle segments with intersection nodes.

The vehicles are assumed to travel at a constant speed and can stop only at the ends of the guide path segments. the guide path network is bidirectional and the vehicles can travel forward or backward. The unit load is one pallet. the number of available vehicle and the duration of loading and unloading at the P/D stations are known.

As many vehicles can travel on the guide path simultaneously, collisions must be avoided. There are two types of collisions: the first type may appear when two vehicles are moving toward the same node. The second type of collision occurs when two vehicles are traveling on a segment in opposite directions.

Every day, a list of orders is given, each order corresponding to a specific product to manufacture (here, a product means one or many units of the same product). Each order determines a sequence of operations on the various cells of the FMS. Then, the production is scheduled. This production scheduling sets the starting time for each order. Pallets of products are moved between the cells by the AGVs. Hence, each material handling request is composed of a pick-up and a delivery with their associated earliest times. At each period, the position of each vehicle must be known. Time is in fifteen second periods.

A production delay is incurred when a load is picked up or delivered after its planned earliest time. The problem is thus defined as follows:

> *Given a number of AGVs and a set of transportation requests, find the assignment of the requests to the vehicles and conflict free routes for the*

vehicles in order to minimize the sum of production delays.

A.1.1 Literature Review

For a recent general review on AGVs problems and issues, the reader is referred to the survey of Qiu *et al.* (2002). These authors identifed three types of algorithms for AGVs problems:

- algorithms for general path topology.
- algorithms for path layout optimization.
- algorithms for specific path topologies.

In our study, we work on algorithms for the general path topology. Methods of this type can be divided in three categories: (1) static methods, where an entire path remains occupied until a vehicle completes its route; (2) time-window based methods, where a path segment may be used by different vehicles during different time-windows; (3) dynamic methods, where the utilization of any segment of path is dynamically determined during routing rather than before routing as with categories (1) and (2).

Our method belongs to the the third category and we focus on bidirectional networks and conflict free routing with an optimization approach. Futhermore we have a static

job set, i.e. all jobs are known *a priori*. Krishnamurthy *et al.* (1993) proposed first and optimization approach to solve a conflict free routing routing problem. Their objective was to minimize the makespan. They assumed that the assignment of tasks to AGVs is given and they solved the routing problem by a column generation method. their method generated very good solutions in spite of the fact that it was not optimal (column generation was performed at the root node of the search tree only). Langevin *et al.* (1996) proposed a dynamic programming based method to solve exactly instances with two vehicles. They solved the combined problem of dispatching and conflict free routing. Desaulniers *et al.* (2003) proposed an exact method and present tests with up to four vehicles. They used slightly the same data set as Langevin *et al.* (1996). Their approach combines a greedy search heuristic (to find a feasible solution and a set of bounds on delays), a column generation method and a branch-and-cut procedure. Their method presents however some limits since its efficiency depends highly on the performance of the starting heuristic. If no feasible solution is found by the search heuristic, then no optimal solution can be found. This search heuristic performs poorly when the level of congestion increases and the system considers at most four AGVs.

A.1.2 A Constraint Programming / Mixed Integer Programming approach

The decisions for dispatching and conflict free routing of automated guided vehicles can be decomposed into two parts: first, the assignments of requests to vehicles with the associated schedule, then the simultaneous routing of every vehicle. The hybrid approach presented herein combines a Constraint Programming (CP) model for the assignment of requests to the vehicles with their actual pick-up or delivery times (in order to minimize the delays) and a Mixed Integer Programming (MIP) model for the conflict free routing. The two models are embedded in an iterative procedure as shown in figure 4.1.

For each assignment and schedule found by the CP model, the MIP model tries to find conflict free routes satisfying the schedule. CP is used to deal with the first

part because it is very efficient for scheduling and, in the present case, it allows identifying easily all optimal solutions. Here optimal solutions are equivalent in terms of value but represent different assignment might yield very different routing solution. The routing part is addressed with MIP since it can be modeled with a time-space network with some interesting sub-structures that allow fast solutions.

The method can be described in three steps:

- Step 1: find an optimal solution $x^\star$ (*i.e.* an assignment of requests to vehicles) to the CP model. Let $z^\star$ the optimal objective function value (the total delay).

- Step 2: use $x^\star$ in the MIP model to find a conflict free routing. If there exist any, the optimal solution to the entire model is found. Otherwise (no feasible solution found), go to step 3.

- Step 3: find another optimal solution to the CP model different from $x^\star$ but with the same objective function value. If there exists any, return to step 2. If no feasible solution has been found with any of the optimal solutions of the CP model, go to step 1 and add a lower bound to the objective function ($f(x) > z^\star$) before solving anew the CP model. This lower bound is set to $z^\star$ and is always updated when returning to step 1.

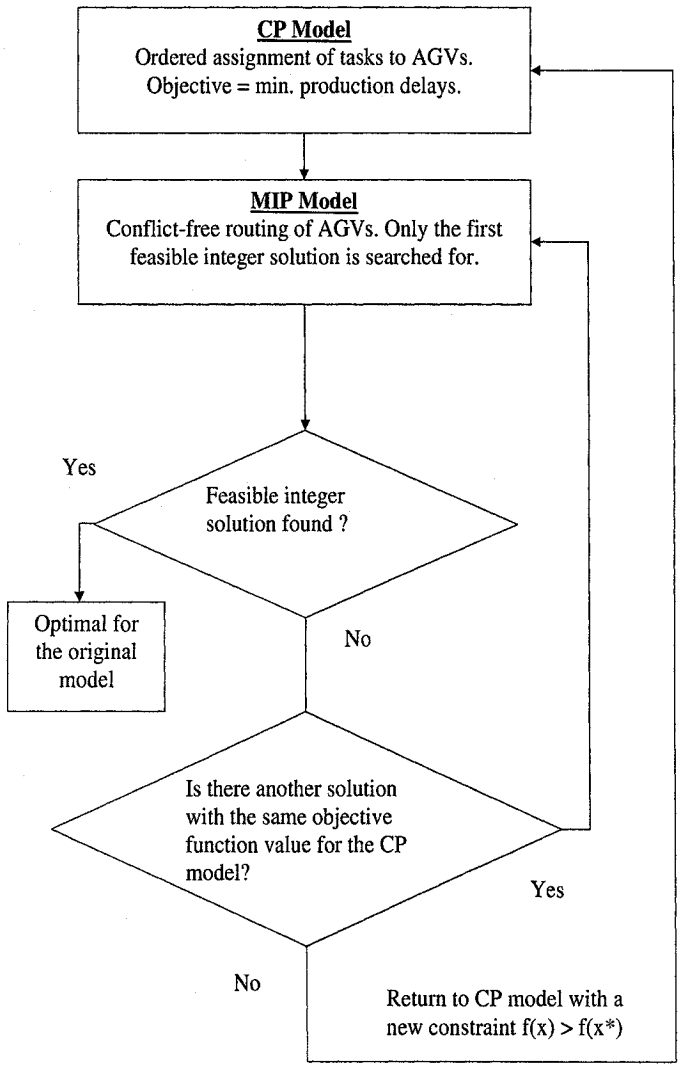

Figure A.1: Decomposition method

A.1.3 The CP model

The model answers to the question to know which vehicle is processing what material handling task and when, by yielding and ordered assignment of tasks to AGVs. The total amount of delays is measured by summing the difference between the actual start time and the earliest start time of all deliveries. In this mmodel, the distance (time) matrix is obtained by using shortest paths between nodes. thus, the delays calculated (which don't take into account the possible conflicts) are an approximation (a lower bound) of the actual delays.

Sets and parameters used:

$DummyStartTasks$:	set of dummy starting tasks. each of them is in fact the starting node of a vehicle corresponding to the last delivery node of a vehicle in the previous planning horizon.
$Start[k]$:	starting node of AGV k.
$Pickups$:	set of pick-up tasks.
$SP[.,.]$:	length of the shortest path between a couple of nodes.
$Node(p)$:	node for task p. It is used here to alleviate the notation.
$nbRequests$:	number of requests to perform.
$nbChar$:	number of vehicles available.
$Requests$:	set of transportation requests. Each request contains two fields: the pick-up task and the associated delivery task
$DummyStartRequests$:	set of dummy starting requests.
$Inrequest$:	set of dummy start requests and real requests.
$Pick[.]$:	Pick-up field of a transportation request.
$Del[.]$:	Delivery field of a transportation request.
$Duration$:	duration of a task.
$Priorities$:	set of couple of tasks linked by a precedence relationship (the first task is to be performed before the other.
$Tasks$:	set of all tasks tasks with a (mandatory) successor.

This model uses the following variables:

$Alloc[i] = k$	if task k is performed by vehicle k. The index lower than 1 represent dummy requests
$Succ[u] = v$	if request v is the successor of request u on the same vehicle.
$Starttime[j]$	is the start time of task j.

For each vehicle, a couple of dummy tasks are created, a tarting task and an end task. The starting task has the following characteristics: its node is the starting node of the AGV, its duration and earliest starting time are set to zero. We define the set **Tasks** by the set of dummy start tasks and real pick-up and delivery tasks. a request consists of a pick-up and a delivery tasks. the constraints used in the model are the following:

$$Alloc[1-k] = Alloc[nbRequests + k] \quad \forall\, k \in Char \tag{A.1}$$

$$\sum_{s \in Inrequest} (Succ[r] = s) = 1 \;\; \forall\, r \in DummyStartRequests \tag{A.2}$$

$$Alloc[o] = Alloc[Succ[o]] \quad \forall\, o \in Tasks \tag{A.3}$$

$$Starttime[d] = 0 \quad \forall\, d \in DummyStartTasks \tag{A.4}$$

$$\begin{aligned}&(Alloc[d] = k) \wedge (Succ[d] = r) \Rightarrow \\ &\quad (Starttime[pick[r]] \geq SP[Start[k], node[pick[r]]) \\ &\qquad \forall\, k \in Char, d \in DummStartRequests, r \in Requests\end{aligned} \tag{A.5}$$

$$Alldifferent(Succ) \tag{A.6}$$

$$Starttime[pick[r]] + 1 + SP[Start[k], node[pick[r]] \leq Starttime[del[r]]$$
$$\forall r \in Requests \tag{A.7}$$

$$Succ[del[r1]] = pick[r2] \Rightarrow$$
$$(Starttime[del[r1]] + 1 + SP[node[del[r1]], node[pick[r2]] \leq Starttime[pick[r2]])$$
$$\forall r1, r2 \in Requests \tag{A.8}$$

$$Starttime[before[u]] + duration[before[u]] \leq Starttime[after[u]]$$
$$\forall u \in Priorities \tag{A.9}$$

$$(Starttime[i] \geq Starttime[j] + 1) \vee$$
$$(Starttime[i] + 1 \leq Starttime[j])$$
$$\forall i, j \in Tasks : (i \neq j) \wedge (node[i] = node[j]) \tag{A.10}$$

Constraints (A.1) ensure that a dummy starting task task and its dummy end task are performed by the same AGV. Constraints (A.2) ensure that the successor of a dummy start request is either a real real request or a dummy end request (in this case the vehicle is idle during the entire horizon but can move to avoid collisions). Constraints (A.3) ensure that every request and its successor must be performed by the same AGV. Constraints (A.4) ensure that, at the beginning of the horizon (period zero), each vehicle is located at its starting node. Constraints (A.5) specify that each vehicle must have enough time to reach its first pick-up node. Constraints (A.6) imply that the successor of each request is unique. Constraints (A.7) specify that each vehicle processing a request must have enough time to go from the pick-up node to the delivery node of the request. Constraints (A.8) ensure that if one request is the successor of another request on the same vehicle, the AGV must have enough time to make the trip from the delivery node of the first request to the pick-up node of the second request. They link the tasks that must be processed at the same nodes so that there is no overlapping. Constraints (A.9) enforce that, for every couple of tasks linked by precedence relationship, the first task must start and be processed before the beginning of the second task. Constraints (A.10) ensure that for each couple of

tasks that must be performed on the same node, one must start one period after the beginning of the other.

A.1.4 The MIP model

For a given schedule obtined from the CP model, the MIP model allows to find whether there exists a feasible routing without conflict. This could be seen as a Constraint Satisfaction Problem since we only search for a feasible routing without conflict. However, the inherent network structure of the routing problem allows using a MIP model where only the first feasible integer solution solution is searched for, thus preventing a potentially time consuming search for the optimal solution of the MIP. The MIP corresponds to a time-space network which defines the position of every vehicle anytime (see Figure 4.2). The original guide path network is composed of segments of length 1, 2 and 3.This network has been transformed into a directed network where all arcs are of length 1 by incorporating dummy nodes on segments of length 2 or 3. At every time period, there is a node for each intersection node (including the dummy nodes) of the guide path. An arc is defined between two nodes of successive time periods if the corresponding intersection nodes are adjacent on the guide path layout. Each vehicle enters the network at a given node at period 0. the time-space network model has the following characteristics:

- One unit of flow is equivalent to one vehicle present in the system.

- The total amount of entering flow at each period is equal to the total number of AGVs (busy or idle).

- At most one unit of flow can enter in a node (no collision can occur at a node).

- There is flow flow conservation at each node.

- An arc whose origin and destination are the same node at two successive periods corresponds to waiting at that node.

- A vehicle can move without having a task to perform, just for avoiding conflicts.

(See Figure 4.2 for the time-space network description).

Several versions of MIPs are presently under investigation. Here, we present on that has given interesting results up to now.

Sets and parameters of the MIP model:

$Char$:	The set of AGVs.
$Nodes$:	The set of nodes.
$Periods$:	The set of periods.
$ArcsPlus$:	The set of all arcs (including those with dummy nodes), represented as an interval of integers).
M :	The length of the horizon.

The variables $Alloc[.]$ and $Starttime[.]$ obtained from the CP model are used as input. $Segment[a]$ is a record having two fields: The first one ($Segment[a].orig$) is the origin of arc a whereas the second field ($Segment[a].dest$) is the destination of a.

The variables of the MIP model are:

$Y[k,t,p]=1$	if vehicle $k \in Char$ is on node $p \in Nodes$ at period $t \in Periods$.
$Z[k,t,a]=1$	if vehicle $k \in Char$ starts visiting arc $a \in ArcsPlus$ at period $t \in [0,..,M-1]$.

The MIP model is defined as follows:

$$Min \sum_{k \in Char,\, t \in Periods,\, p \in Nodes} Y[k,t,p]$$

s.c.

$$Y\left[k, Starttime\left[2-k\right], node\left[Task\left[2-k\right]\right]\right]=1 \qquad \forall\, k \in Char \tag{A.11}$$

$$Y\left[Alloc\left[r\right], Starttime\left[pick\left[r\right]\right], node\left[Task\left[r\right]\right]\right]=1 \qquad \forall\, r \in Requests \tag{A.12}$$

$$Y\left[Alloc\left[r\right], Starttime\left[del\left[r\right]\right], node\left[Task\left[r\right]\right]\right]=1 \qquad \forall\, r \in Requests \tag{A.13}$$

$$Y\left[Alloc\left[r\right], Starttime\left[pick\left[r\right]\right]+1, node\left[Task\left[r\right]\right]\right]=1 \qquad \forall\, r \in Requests \tag{A.14}$$

$$Y\left[Alloc\left[r\right], Starttime\left[del\left[r\right]\right]+1, node\left[Task\left[r\right]\right]\right]=1 \qquad \forall\, r \in Requests \tag{A.15}$$

$$\sum_{p \in Nodes} Y\left[k,t,p\right]=1 \qquad \forall\, t \in Periods, k \in Char \tag{A.16}$$

$$Y\left[k,t,Segment\left[a\right].orig\right]+Y\left[k,t+1,Segment\left[a\right].dest\right]-Z\left[k,t,a\right] \leq 1$$
$$\forall\, k \in Char, t \in [0,..,M-1], a \in Arcs \tag{A.17}$$

$$Z[k,t,a] \leq Y[k,t,Segment[a].orig]$$
$$\forall k \in Char, t \in [0,..,M-1], a \in Arcs \quad \text{(A.18)}$$

$$Z[k,t,a] \leq Y[k,t+1,Segment[a].dest]$$
$$\forall k \in Char,\ t \in [0,..,M-1], a \in Arcs \quad \text{(A.19)}$$

$$\sum_{a \in ArcsPlus} Z[k,t,a] = 1$$
$$\forall t \in [0,..,M-1],\ k \in Char \quad \text{(A.20)}$$

$$\sum_{p \in Nodes} Y[k,t,p] \leq 1 + \left(\sum_{r \in Realtasks : t=Starttime[r] \wedge p = node[Task[r]]} 1 \right) * \left(\sum_{r \in Realtasks : t=Starttime[r]-1 \wedge p = node[Task[r]]} 1 \right)$$
$$\forall t \in Periods,\ k \in Char \quad \text{(A.21)}$$

$$\sum_{k \in Char} Z[k,t,a] + \sum_{k \in Char,\, b \in Opp[a]} Z[k,t,b] \leq 1$$
$$\forall k \in Char,\ t \in [0,..,M-1],\ a \in Arcs \quad \text{(A.22)}$$

Constraints (A.11) specify that every vehicle must be present at its starting node at period 0. Constraints (A.12-A.13) enforce the presence of vehicles at their task node in due time. Constraints (A.14-A.15) ensure that every vehicle stays at least one period at its task node to load or unload. Constraints (A.16) unsure that each vehicle has an unique position at each period. Constraints (A.17) imply that if a vehicle starts visiting the origin of an arc at period t, it will visit the destination at period $t+1$. Constraints (A.18) enforce that if a vehicle is on a node at period t, it means that it has started visiting an incoming arc (waiting arc or not) at period $t-1$. Constraints (A.19) enforce that if a vehicle avehicle is on a node at period $t+1$, it means that it has started visiting an incoming arc (waiting arc or not) at period t. Constraints (A.20) ensere that each vehicle starts running on an unique arc (weal or

waiting arc) at each period. Constraints (A.21) forbid the presence of two vehicles on the same node except the case where one vehicle is finishing its task while another one is starting its task on a work station. in a certain sense, these are anti collision constraints on nodes. Constraints (A.22) are anti-collision constraints on arcs: no two vehicles travel at the same time on the same arc in opposite directions.

A.2. Preliminary Results

The method has been implemented in OPL Script. We compared our method to the approach of (Desaulniers *et al.* (2003)) and we not only gain on flexibility by using CP but we solved new problems with five or six AGVs (Desaulniers *et al.* (2003) displayed results with up to four AGVs). The number of AGVs used limited to six since it didn't make sense to increase the number of AGVs with regards to the number of requests. However, larger applications like container terminal operations use dozens of AGVs and no optimization automated solution approach exist. Presently, the size of the MIP model for the routing part is very large. It depends largerly on the size of the horizon. then larger time horizons will likely be more difficult to handle. we need to test our method on problems of larger number of tasks or AGVs with the idea of rolling horizons.

Our method took more computing time than that of Desaulniers *et al.* (2003) even though the computation times found are below the limits of ten minutes that we set. our tests were done on a Pentium 4, 2.5 GHz. Desaulniers *et al.* (2003) did their tests on a SUNFIRE 4800 workstation (900MHz).

Our approach can be transformed into a heuristic version by limiting the time of each scheduling solution to 30 seconds. Experiments are planned to see if this technique can yield quickly very good solutions.

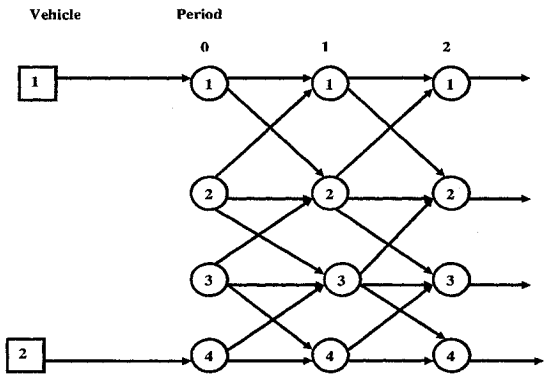

Figure A.2: Description of the time-space network (MIP)

A.3. Conclusion

This article reports on the development of a flexible hybrid algorithm based on the decomposition into CP and MIP component. Tests on problems with a greater number of tasks or AGVs are needed to fully evaluate the effectiveness of the proposed method. It would be interesting to analyze the impact of the number and the diversity of precedence relationship between tasks. This research is an ongoing project as we are working on refining the presented models and the iterative loop that guides them. as future work, an adaptation of our approach to a mining context should be very interesting due to the high level of congestion present in these problems.

References

DESAULNIERS,G., LANGEVIN, A., RIOPEL, D. AND VILLENEUVE, B. (2003). Dispatching and Conflict-Free Routing of Automated Guided Vehicles : An Exact Approach. *The International Journal of Flexible Manufacturing Systems* **15** 309–331.

KRISHNAMURTHY, N.N., BATTA, R. AND KARWAN, M.H. (1993). Developing Conflict-Free Routes for Automated Guided Vehicles. *Operations Research* **4** 1077–1090.

LANGEVIN, A., LAUZON, D. AND RIOPEL, D. (1996). Dispatching, Routing and Scheduling of Two Automated Guided Vehicles in a Flexible Manufacturing System. *International Journal of Flexible Manufacturing Systems* **8** 246–262.

QIU, L., HSU, W.-J. AND WANG, H. (2002). Scheduling and Routing Algorithms for AGVs : A Survey. *International Journal of Production Research* **40** 745–760.